導入ガイド

iPlanet Directory Server

Version 5.1

816-4124-01
2001 年 12 月

目 次

本書について

iPlanet Directory Server 5.1 は、業界標準の LDAP (Lightweight Directory Access Protocol) に基づく、スケーラブルで強力な分散型ディレクトリサーバです。iPlanet Directory Server は、社内イントラネット、取引先とのエクストラネット、および顧客との窓口となる公共のインターネット上で使用できる、集中・分散型のデータリポジトリを構築するための基盤となります。

このリリースの iPlanet Directory Server の新機能および拡張機能に関する最新情報は、以下のサイトにあるオンラインリリースノートを参照してください。

```
http://docs.iplanet.com/docs/manuals/directory.html
```

このマニュアルの目的

このマニュアルでは、ディレクトリの導入を計画するための基本事項について説明します。ここで提供する情報は、ディレクトリの責任者、設計者、および管理者を対象としています。

最初の章では、ディレクトリの基本概念を紹介します。残りの章では、スキーマの設計、ディレクトリツリー、トポロジ、複製、セキュリティなど、ディレクトリ設計のさまざまな側面について説明します。最後の章では、簡単な構成で Directory Server を導入する際に役立つ導入例を示します。この例は、世界中に分散した数百万のユーザにも対応できる複雑な構成の Directory Server を導入するときにも役立ちます。

表記上の規則

ここでは、このマニュアルで使用している表記上の規則について説明します。

クーリエ (等幅) フォント : このフォントは、テキストに表示される属性およびオブジェクトクラスの名前などの、リテラルテキストに使用します。また、ファイル名、および例の記述にも使用します。

イタリック体 : このフォントは、オブジェクトクラスが継承する属性名およびその説明に使用します。また、実際の値に置き換えるテキスト (パス名の可変部分など) にも使用します。

注	「注」、「注意」、および「ヒント」は、重要な情報や制限を示します。必ずこれらの注意事項を読んでから、次の作業を続けるようにしてください。

このマニュアルでは、パスとファイル名に次の形式を使用しています。

installDir`/slapd-`*serverID*`/...`

実際のパスとサーバ識別子は、プラットフォーム、インストール、および構成によって異なります。デフォルトパスは、プラットフォームによって次のようになります。

Solaris 9 プラットフォーム	`/var/ds5/slapd-`*serverID*`/...`
その他の UNIX プラットフォーム	`/usr/iplanet/servers/slapd-`*serverID*`/...`
Windows プラットフォーム	`C:¥iPlanet¥Servers¥slapd-`*serverID*`¥...`

Directory Server を別の場所にインストールした場合は、それに合わせてパスを変更してください。*serverID* は、サーバのインストール時に指定したサーバ識別子を示します。たとえば、Directory Server に `phonebook` という名前を付けた場合、実際のパスは次のようになります。

Solaris 9 プラットフォーム	`/var/ds5/slapd-phonebook/...`
その他の UNIX プラットフォーム	`/usr/iplanet/servers/slapd-phonebook/...`
Windows プラットフォーム	`C:¥iPlanet¥Servers¥slapd-phonebook¥...`

このマニュアルに記載されている大半のパスとコマンドは UNIX 形式です。Windows ベースの Directory Server を使用する場合は、UNIX 形式のパスとコマンドを Windows 形式のパスとコマンドに読み替えてください。Windows プラットフォームのコマンドには、UNIX と同じコマンド名に拡張子 `.exe` または `.bat` が付きます。

関連情報

iPlanet Directory Server のマニュアルセットには、次のマニュアルも含まれています。

『iPlanet Directory Server インストールガイド』Directory Server のインストール手順と、Netscape Directory Server から iPlanet Directory Server へ移行する手順について説明します。

『iPlanet Directory Server 管理者ガイド』ディレクトリの内容を管理し、Directory Server を保守するための手順と、サーバ側プラグインの設定に関する情報を提供します。

『iPlanet Directory Server 構成、コマンド、およびファイルのリファレンス』Directory Server に付属するコマンド行スクリプトの使用方法について説明します。

『iPlanet Directory Server スキーマリファレンス』Directory Server に含まれている、クライアントアプリケーションで役立つ LDAP スキーマに関する情報を提供します。

その他の有用な情報は、次の Web サイトから入手できます。

- iPlanet 製品のオンラインマニュアル :
  `http://docs.iplanet.com/docs/manuals/`
- iPlanet 製品の技術情報 :
  `http://www.iplanet.com/support/technical_resources/`
- iPlanet プロフェッショナルサービスに関する情報 :
  `http://www.iplanet.com/services/professional_services_3_3.html`
- Solaris 対応 Sun Enterprise Service のパッチとサポート :
  `http://www.sun.com/service/`
- iPlanet の開発者向け情報 :
  `http://developer.iplanet.com/`
- iPlanet のトレーニング情報 :
  `http://www.iplanet.com/learning/index.html`
- iPlanet 製品のデータシート :
  `http://www.iplanet.com/products/index.html`

第 1 章

Directory Server の概要

iPlanet 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、供給業者、提携業者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、ユーザのプロファイルや環境設定、エクストラネットのユーザ認証などを管理することもできます。

この章では、ディレクトリを設計する前に理解しておく必要がある基本事項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。

- ディレクトリサービスとは
- iPlanet Directory Server の概要
- ディレクトリ設計の概要
- その他の一般的なディレクトリ関連資料

ディレクトリサービスとは

ディレクトリサービス (directory service) は、企業や加入者あるいはその両方に関する情報を格納し、その情報をユーザが使用できるようにするための、ソフトウェア、ハードウェア、および処理の集合です。ディレクトリサービスは、少なくとも 1 つの Directory Server のインスタンスと、1 つ以上のディレクトリクライアントプログラムから構成されます。クライアントプログラムは、ディレクトリに格納されている名前、電話番号、住所、およびその他のデータにアクセスできます。

よく知られているディレクトリサービスの 1 つに、DNS (ドメインネームシステム) サーバがあります。DNS サーバは、コンピュータのホスト名を IP アドレス (IP address) に割り当てます。そのため、コンピュータ資源 (ホスト) はすべて DNS サーバのクライアントになります。ホスト名を割り当てることにより、計算資源のユーザは、数値の IP アドレスではなく覚えやすいホスト名を使用して、ネットワーク上のコンピュータを簡単に見つけることができます。

ただし、DNS サーバは名前と IP アドレスの 2 種類の情報しか格納していません。実際のディレクトリサービスでは、格納される情報の種類に制限はありません。

iPlanet Directory Server では、ネットワークを介してアクセスできる 1 つのリポジトリに、すべての情報を格納します。ディレクトリに格納できる情報の例を次に示します。

- 組織のプリンタに関するデータ (設置場所、カラーまたは白黒の区別、製造元、購入日、シリアル番号) などの物理デバイス情報
- 名前、電子メールアドレス、所属部署などの、公開社員情報
- 給与、社会保障番号、自宅住所、電話番号などの、非公開社員情報
- 顧客名、最終納品日、入札情報、契約番号、プロジェクト日程などの、契約または取引の情報

iPlanet Directory Server は、多種多様なアプリケーションの要件に対応しています。また、格納されている情報にアクセスするための標準プロトコルと API (アプリケーションプログラミングインタフェース) も提供しています。

次に、グローバルディレクトリサービスと LDAP (Lightweight Directory Access Protocol) について説明します。

グローバルディレクトリサービスについて

iPlanet Directory Server は、さまざまな種類のアプリケーションに情報を提供するグローバルなディレクトリサービスを提供します。最近まで、多くのアプリケーションはそれぞれ独自の専用データベースと一体になっていました。専用データベースは 1 つのアプリケーションだけを使用している場合には便利ですが、複数のデータベースが同じ情報を管理している場合は、その管理業務が大きな負担となります。

たとえば、それぞれが独自の専用ディレクトリサービスを備えた 3 種類の専用電子メールシステムをネットワークで使用しているとします。1 つのディレクトリでパスワードを変更した場合、その変更が自動的にほかのディレクトリに複製されることはありません。同じ情報を持つ複数のインスタンスの管理は、ハードウェアコストと人的コストの増大を招きます。この問題は、「N + 1 ディレクトリ問題 (n + 1 directory problem)」と呼ばれています。

グローバルディレクトリサービスは、すべてのアプリケーションがアクセス可能なディレクトリ情報の中央リポジトリを 1 つ装備することで、「N + 1 ディレクトリ問題」を解決しています。ただし、多様なアプリケーションがディレクトリにアクセスできるようにするには、アプリケーションとディレクトリ間でネットワークを介した通信手段が必要です。iPlanet Directory Server では、LDAP (Lightweight Directory Access Protocol) を使用して、アプリケーションがグローバルディレクトリサービスにアクセスできるようにしています。

LDAP について

LDAP は、クライアントアプリケーションとサーバが相互通信を行うための共通言語を提供します。LDAP は、ISO X.500 標準で使用される DAP (Directory Access Protocol) の「軽量」バージョンです。DAP では、任意のアプリケーションが、拡張可能で堅牢な情報のフレームワークを通じてディレクトリにアクセスできますが、管理コストがかかりすぎます。また DAP では、インターネット標準の TCP/IP プロトコル以外の通信層が使用されており、ディレクトリの命名規則が複雑です。

LDAP は、DAP の優れた機能を保持しながら、管理コストの削減を図っています。LDAP では、TCP/IP 上で動作する開放型ディレクトリアクセスプロトコルと簡易符号化方式を使用します。LDAP は、X.500 標準 (X.500 standard) のデータモデルを維持しながら、ハードウェアとネットワークインフラストラクチャに適度な投資をすれば、数百万のエントリをサポートすることが可能です。

iPlanet Directory Server の概要

iPlanet Directory Server には、ディレクトリそれ自体である LDAP プロトコルを実装するサーバ側のソフトウェア、およびエンドユーザによるディレクトリ内のエントリの検索と変更を可能にするグラフィカルユーザインタフェースが含まれます。iPlanet Console のディレクトリマネージャや Netscape Communicator 4.x のアドレス帳など、ほかの LDAP クライアントも使用できます。さらに、ほかの LDAP クライアントプログラムを購入したり、iPlanet Directory Server 製品に含まれている LDAP クライアント SDK を使用して独自のプログラムを作成したりすることもできます。

ほかの LDAP クライアントプログラムを追加しなくても、Directory Server はインターネットやエクストラネット向けの基盤を提供します。すべての iPlanet サーバは、社員や顧客、供給業者や提携業者に関するデータなど、共有サーバ情報用の中央リポジトリとしてディレクトリを使用します。

Directory Server を使用すると、エクストラネットのユーザ認証の管理、アクセス制御の作成、ユーザ環境の設定、ユーザの集中管理などを行うことができます。ホストされる側の環境では、提携業者、顧客、供給業者が、それぞれに関連するディレクトリ部分を管理できるので、管理コストを削減できます。

Directory Server のインストール時に、使用しているマシンに次のコンポーネントがインストールされます。

- LDAP サーバ (Directory Server) とプラグインインタフェース

  このプロセスの名前は `ns-slapd` です。

- iPlanet Administration Server

  Administration Server については、
  http://iplanet.com/products/iplanet_application/ を参照してください。

- サーバ管理用の iPlanet Console

  iPlanet Console については、
  http://docs.iplanet.com/docs/manuals/console.html にある Console のマニュアルを参照してください。

- サーバの起動と停止、データベースのデータのインポートとエクスポート、データベースのインデックスの再作成、アカウントの有効化と無効化、LDIF のマージ、カーネルのチューニングなどを行うためのコマンド行ツール

  コマンド行ツールについては、『iPlanet Directory Server 構成、コマンド、およびファイルのリファレンス』を参照してください。

- SNMP モニター

  SNMP 監視機能については、『iPlanet Directory Server 管理者ガイド』を参照してください。

このマニュアルでは、中核となる Directory Server と、Directory Server での処理に使用されるプラグインについて説明します。次の節では、Directory Server について詳しく説明します。説明する内容は以下のとおりです。

- 16 ページの「Directory Server アーキテクチャの概要」
- 19 ページの「Directory Server のデータ保存」

Directory Server アーキテクチャの概要

Directory Server には、インストール時点で次のコンポーネントが含まれています。

- ネットワーク通信を実行するサーバフロントエンド
- アクセス制御や複製などを行うためのサーバ機能のプラグイン

- サーバに関連するデータを格納する基本ディレクトリツリー

次に、ディレクトリの各コンポーネントについて詳しく説明します。

サーバフロントエンドの概要

Directory Server のサーバフロントエンドは、ディレクトリクライアントプログラムとの通信を管理します。Directory Server は、UNIX システム上ではデーモン、Windows NT と Windows 2000 上ではサービスとして機能します。複数のクライアントプログラムが、LDAP を通じてサーバと対話できます。これらのクライアントプログラムは、TCP/IP 上の LDAP を使用して通信できます。クライアントが接続に TLS (Transport Layer Security) を使用する場合は、SSL/TLS で接続を保護することもできます。

通常、TLS で通信を行う場合は通信が暗号化されます。将来、DNS セキュリティが実装されている場合は、安全な DNS と組み合わせて使用される TLS によって、クライアントアプリケーションが適切なサーバに接続しているかどうかクライアントアプリケーションに通知されるようになります。クライアントに証明書が発行されている場合、iPlanet Directory Server では TLS を使用して、クライアントにサーバへのアクセス権があることを確認できます。TLS およびその前身である SSL は、iPlanet Directory Server の製品で、メッセージの整合性検査、デジタル署名、サーバ間の相互認証などのセキュリティ機能を実行するために使用されます。

Directory Server はマルチスレッドアプリケーションなので、複数のクライアントが同じネットワーク上で同時にサーバに接続することができます。ディレクトリサービスが、膨大な数のエントリや物理的に分散している多数のクライアントを取り込んでその規模を拡大するにつれて、複数の Directory Server をネットワーク上の戦略的な位置に配置するようにします。

サーバプラグインの概要

Directory Server はプラグインを利用しています。プラグインは、コアサーバの機能を拡張する手法の 1 つです。たとえば、プラグインの例として、データベースが挙げられます。

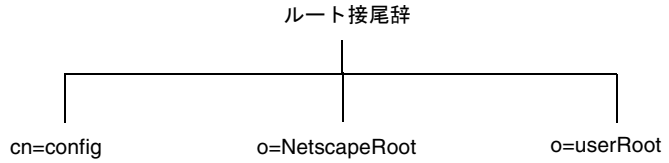
プラグインは、無効にすることもできます。無効にすると、プラグインの構成情報はディレクトリ内に残りますが、その機能はサーバで使用されなくなります。ディレクトリでどのような処理をするかに応じて、Directory Server が提供するプラグインの中から利用したいプラグインを選択できます。

iPlanet プロフェッショナルサービスでは、Directory Server の導入においてカスタムプラグインを作成するサービスを提供しています。詳細は、iPlanet プロフェッショナルサービスにお問い合わせください。

基本ディレクトリツリーの概要

ディレクトリツリー (directory tree) はディレクトリ情報ツリーまたは DIT とも呼ばれ、ほとんどのファイルシステムで使用されるツリーモデルに酷似して、階層の最上位にツリーのルート、つまり最初のエントリが置かれています。インストール時に、Directory Server によってデフォルトのディレクトリツリーが作成されます。

デフォルトのディレクトリツリーを次に示します。



ツリーのルートはルート接尾辞 (root suffix) と呼ばれます。ルート接尾辞の命名方法については、58 ページの「接尾辞の選択」を参照してください。

インストール時に、ディレクトリにはルート接尾辞の下に最大 4 つのサブツリーを作成できます。

- `cn=config`

  このサブツリーには、サーバの内部構成に関する情報が含まれます。

- `o=NetscapeRoot`

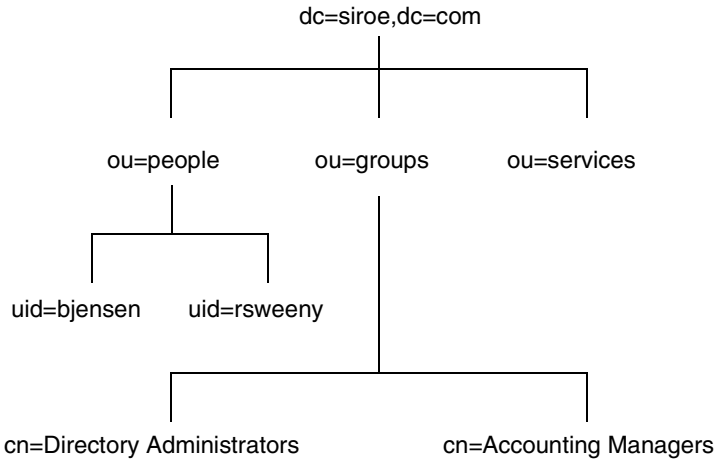
  このサブツリーには、iPlanet Administration Server など、ほかの iPlanet サーバの構成情報が含まれます。Administration Server は、認証のほかに、LDAP 経由では実行できないすべての処理 (起動や停止など) を行います。

- `o=userRoot`

  インストール時に、ユーザデータベースがデフォルトで作成されます。このデータベースのデフォルト名は、`o=userRoot` です。このデータベースは、インストール時に生成することも、あとで生成することもできます。

注 Directory Server の別のインスタンスをインストールするときに、`o=NetscapeRoot` 情報を含めずに、別のサーバにある構成ディレクトリまたは `o=NetscapeRoot` サブツリーを使用するように指定できます。構成ディレクトリの位置の決定方法については、『iPlanet Directory Server インストールガイド』を参照してください。

デフォルトのディレクトリツリーを基に、インストールするディレクトリに関連するデータを追加できます。次に、siroe.com 社のディレクトリツリーの例を示します。



ディレクトリツリーの詳細については、第 4 章「ディレクトリツリーの設計」を参照してください。

Directory Server のデータ保存

ディレクトリデータは、LDBM データベース (LDBM database) に保存されます。LDBM データベースは、ディレクトリとともに自動的にインストールされるプラグインとして実装されており、デフォルトで有効になっています。

データベースは、保存、性能、複製、およびインデックス付けに適用される基本単位です。データベースに対して、インポート、エクスポート、バックアップ、復元、インデックス付けなどの操作を実行できます。

デフォルトでは、Directory Server はディレクトリツリーを 1 つのデータベースに収めます。このデータベースにより、数百万のエントリを管理できます。デフォルトデータベースには、データを安全に保護するために、データのバックアップと復元を行うための高度なツールが用意されています。

複数のデータベースを使用して Directory Server をサポートするように選択できます。データを複数のデータベースに分散すると、1 つのデータベースに保存するよりも多くのデータをサーバで保持できます。

次に、ディレクトリデータベースによるデータの保存方法について説明します。

ディレクトリのエントリについて

LDIF (LDAP Data Interchange Format) は、ディレクトリのエントリを記述するための標準のテキスト形式です。エントリ (entry) は、組織のメンバーやネットワーク上のプリンタなどのオブジェクトに関する情報を含む LDIF ファイルの行のグループです。エントリについての情報は、LDIF ファイル内で属性や値のセットで表されます。各エントリはオブジェクトの種類を示すオブジェクトクラス (object class) 属性を持ち、このオブジェクトに含まれている属性をエントリで記述して定義します。各属性 (attribute) は、エントリの特性を示します。

たとえば、オブジェクトクラスが `organizationalPerson` で、特定の組織のメンバーを表すエントリがあるとします。このオブジェクトクラスには、`givenname` 属性と `telephoneNumber` 属性があります。これらの属性に割り当てられた値は、エントリが示すメンバーの名前と電話番号を表します。

iPlanet Directory Server は、サーバによって計算される読み取り専用属性も使用します。これらの属性は、操作属性と呼ばれます。アクセス制御やその他のサーバ機能に関連した、管理者が設定可能な操作属性もあります。

エントリは、ディレクトリツリー内の階層構造に保存されます。LDAP では、1 つのエントリを照会し、ディレクトリツリー内でそのエントリの下位にあるすべてのエントリを要求できます。このサブツリーは、ベース識別名またはベース DN と呼ばれます。たとえば、`ou=people, dc=siroe,dc=com` というベース DN を指定して LDAP 検索を要求すると、`dc=siroe,dc=com` ディレクトリツリーの `ou=people` サブツリーだけが検索されます。

ただし、LDAP 検索に呼応してすべてのエントリが自動的に返されるわけではありません。`ldapsubentry` オブジェクトクラスのエントリは、通常の検索要求では返されません。`ldapsubentry` エントリは管理オブジェクトを表します。たとえば、ロール (role) やサービスクラス (Class of Service) を定義するエントリは、Directory Server で内部的に使用されます。これらのエントリが返されるようにするには、クライアントは特別な方法で `ldapsubentry` オブジェクトクラスのエントリを検索する必要があります。

ロールについては、71 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。サービスクラスについては、73 ページの「サービスクラス」を参照してください。

ディレクトリデータの分散

ツリーのさまざまな部分を別々のデータベースに格納すると、ディレクトリがクライアントの要求を並行して処理できるようになり、性能が向上します。データベースを異なるマシンに保存すれば、さらに性能を向上させることができます。

分散されたデータに接続するには、ディレクトリのサブツリーに特殊なエントリを作成します。このエントリの下で実行されるすべての LDAP 操作は、エントリが実際に格納されているリモートマシンに送信されます。この手法は、連鎖 (chaining) と呼ばれます。

連鎖はプラグインとしてサーバに実装されます。プラグインはデフォルトで有効になっています。このプラグインを使用して、離れた場所に格納されているデータをポイントするための特殊なエントリである、データベースリンクを作成できます。クライアントアプリケーションがデータベースリンク (database link) からデータを要求すると、データベースリンクはリモートデータベースからデータを検索して、クライアントに返します。

ディレクトリ設計の概要

前節では、ディレクトリサービスの概要を説明し、そして iPlanet Directory Server について特に説明しました。ここでは、ユーザ独自のディレクトリサービスの設計について説明します。

実際にディレクトリサービスを導入する前に設計計画を立てることは、ディレクトリを確実に成功させるためのもっとも重要な作業です。ディレクトリの設計時には、環境、データソース、ユーザ、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。これらのデータを活用することにより、要件を満たすディレクトリサービスを設計できます。

ただし、iPlanet Directory Server が備える柔軟性により、Directory Server を導入した後も、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。

設計プロセスの概要

次に、設計プロセスを 6 段階に分けて説明します。

- ディレクトリデータの設計

  ディレクトリには、ユーザ名、電話番号、ユーザ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設計」を参照してください。第 2 章では、ディレクトリに保存するデータのタイプや、Directory Server の内容の設計に必要なほかの作業について説明しています。

- スキーマの設計

  ディレクトリは、1 つ以上の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します。これらのアプリケーションには、ディレクトリに格納するデータについて、形式などの要件があります。ディレクトリに格納するデータの特性は、ディレクトリスキーマで決定します。第 3 章「スキーマの設計」では、iPlanet 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ています。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。ディレクトリツリーの目的は、この編成と参照です。第 4 章「ディレクトリツリーの設計」では、ディレクトリツリーを紹介し、データ構造の設計方法を説明しています。ディレクトリツリーの設計例も記載されています。

- ディレクトリトポロジの設計

  トポロジの設計では、ディレクトリツリーを複数の物理的な Directory Server 上に分割する方法や、これらのサーバ間での通信方法を決定します。第 5 章「ディレクトリトポロジの設計」では、トポロジ (topology) 設計の基礎となる一般原則、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するメカニズム、ディレクトリ自体で分散したデータを追跡する方法について説明しています。

- 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、性能が向上し、耐障害性が高まります。第 6 章「レプリケーションの設計」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ています。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ています。

- 安全なディレクトリの設計

  最後に、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、サービスをほかの側面から検討します。第 7 章「安全なディレクトリの設計」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ています。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ています。

ディレクトリの導入

ディレクトリサービスの設計が完了したら、導入フェーズに進みます。導入フェーズは、次のステップから構成されます。

- ディレクトリの試験
- ディレクトリの実際の導入

ディレクトリの試験

導入フェーズの最初のステップでは、サーバ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導入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。

ディレクトリの実際の導入

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導入から実際の導入に移す計画を立て、それを実行します。次のような内容の導入計画を作成します。

- 必要な資源の見積もり
- サーバをインストールする前にやっておくべき作業
- 達成すべき事柄と作業日程
- 導入が成功したかどうかを測定するための一連の基準

ディレクトリサービスのインストールについては、『iPlanet Directory Server インストールガイド』を参照してください。ディレクトリの管理と保守については、『iPlanet Directory Server 管理者ガイド』を参照してください。

その他の一般的なディレクトリ関連資料

ディレクトリ、LDAP、および LDIF については、次のドキュメントを参照してください。

- RFC 2849: The LDAP Data Interchange Format (LDIF) Technical Specification
  http://www.ietf.org/rfc/rfc2849.txt
- RFC 2251: Lightweight Directory Access Protocol (v3)
  http://www.ietf.org/rfc/rfc2251.txt
- 『Understanding and Deploying LDAP Directory Services』
  (T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年)

第 2 章

ディレクトリデータの設計

ディレクトリに格納されるデータには、ユーザ名、電子メールアドレス、電話番号、ユーザが所属するグループの情報など含まれます。ほかのタイプ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、およびアクセスの要求方法と応答方法が決まります。

この章では、ディレクトリデータを計画するときの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
- ディレクトリ要件の定義
- サイト調査の実施

ディレクトリデータの概要

データのタイプによっては、ディレクトリに適した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チューニングされているため、書き込み操作はサーバの性能を低下させます。

- 属性 - データの形式で表現可能 (たとえば、`surname=jensen`)
- 多くのユーザ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、多くのユーザやアプリケーションにとって重要です。

- 複数の場所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明細書情報
- 顧客名、期限、作業内容、価格情報などの契約情報
- ユーザのソフトウェアの環境設定情報や構成情報
- Web サーバへのポインタ、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資源の場所

サーバ管理以外でも Directory Server を使用する場合は、ディレクトリにどのような情報を格納するかを決定する必要があります。たとえば、次のタイプの情報を含める場合もあります。

- 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ことは得意ですが、画像やほかのメディアなどの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、ディレクトリには FTP、HTTP、またはほかのタイプの URL 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格納することはできます。

ディレクトリ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。ディレクトリで実行される書き込み操作の回数を減らすことにより、検索処理全体の性能が向上します。

ディレクトリ要件の定義

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決定しておきます。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、ディレクトリデータを拡張したり分散させる際影響します。

設計時には、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。ディレクトリの導入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入れるか。たとえば、使用している会計パッケージが現時点では LDAP に対応していないが、近い将来 LDAP に対応することがわかっている場合など。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する
- 将来ディレクトリに格納する可能性のあるデータがあるか。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る。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る。これは予想される中でもっとも困難な場合であるが、考慮しておけば予想しなかった事態に対処できる。少なくとも、このような方法で計画を行っていれば、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つ

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。ディレクトリに入れるデータはディレクトリのアーキテクチャにとってもっとも重要です。このため、サイト調査には十分時間をかけてください。サイト調査では、次のような作業を実施します。ここでは各作業を簡単に説明し、それから詳しく解説します。

- ディレクトリを使用するアプリケーションを特定する

  導入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。

- 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 NT、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 (ユーザまたはグループなど) と、ディレクトリ内で管理するオブジェクトの属性 (ユーザ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、データの複製方法や、リモートサーバ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構成が変わってきます。

  レプリケーションについては、99 ページの第 6 章「レプリケーションの設計」を参照してください。連鎖については、79 ページの第 5 章「ディレクトリトポロジの設計」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、負荷均衡と回復の目的のために、ほかのサーバ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の両方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一か所に配置し、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低減できます。

- 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導入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。

ディレクトリ対応のアプリケーションの特定

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、次のようなものがあります。

- オンラインの電話帳などのディレクトリブラウズアプリケーション。ユーザが必要とする情報 (電子メールアドレス、電話番号、社員名など) を決定し、それらの情報がディレクトリに含まれるようにする
- 電子メールアプリケーション、特に電子メールサーバ。すべての電子メールサーバで、ディレクトリで利用可能な電子メールアドレス、ユーザ名、およびルーティング情報が必要。ただし、サーバの中には、ユーザ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る
- 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では、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、役職など、個人に関する詳細な情報が必要

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	名前、電子メールアドレス、電話番号、ユーザ 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	ユーザ、グループ	ユーザ ID、パスワード、グループ名、グループ番号、グループの所有者
Calendar Server	ユーザ、会議室	名前、ユーザ ID、広さ、会議室の名前

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、ディレクトリを利用するアプリケーションが必要とするデータの特定に役立ちます。

ディレクトリで管理するデータのタイプと、そのデータの管理を開始する時期についての最終的な決定は、次の要因に影響されます。

- さまざまな古い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の数
- 古い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

ディレクトリに入れるすべてのデータを調べるには、現存のデータの格納場所について、次のような調査を実施する必要があります。

- 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、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 NT、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

  データの管理を集中化するときに、新しいツールや新しい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	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、グループ	名前、電子メールアドレス、ユーザ ID、パスワード、電子メールの環境設定

表 2-2　情報ソース (続き)

データソース	データクラス	データ
設備システム	施設	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

ディレクトリに含めるために特定したすべてのデータは、次のような一般的な観点から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する必要があります。これにより、第 3 章「スキーマの設計」で詳しく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	ユーザのエントリ
ファックス番号	電話番号	14 桁の数字	施設	ユーザのエントリ
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	ユーザのエントリ

サービスレベルの決定

提供するサービスレベルは、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が必要とするサービスによって決まります。各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、通常の運用レベルからミッションクリティカルなレベルまで、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。ディレクトリの導入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。

たとえば、システム全体に及ぶような障害が発生する可能性を排除する場合は、同じデータに対して複数のマスターが存在する、マルチマスター構成を使用します。次に、データマスターの特定について詳しく説明します。

データマスターの特定

データマスター (data master) は、データのマスターソースになるサーバです。データが複数の場所に存在する場合は、どのサーバをデータマスターにするかを考慮します。たとえば、複製を使用する場合、あるいは LDAP 経由で通信できないアプリケーションを使用する場合は、データが複数のサイトに分散されていることがあります。あるデータが複数の場所に存在する場合は、マスターコピーを置くサーバとこのマスターコピーから更新を受け取るサーバを決定しておく必要があります。

レプリケーション時のデータマスターの作成

iPlanet Directory Server では、複数のサーバに情報のマスターソースを置くことができます。レプリケーションを使用する場合は、どのサーバをデータのマスターソースにするかを決定します。iPlanet Directory Server では、複製のサーバが同じデータのマスターソースとなる可能性があるマルチマスター構成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 (multi-master replication) については、99 ページの「レプリケーションの設計」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 台の 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 (consumer) サーバにレプリケートするようにします。2 台のマスターサーバがあれば、1 台のサーバが故障してオフラインになったときでも安全が保障されます。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

  多重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、一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、iPlanet MetaDirectory 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。iPlanet MetaDirectory については、iPlanet の販売代理店に問い合わせるか、http://www.iplanet.com/ にある iPlanet の Web サイトをご覧ください。

- 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する

  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。

たとえば、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管理データベースの両方に格納されます。人事管理アプリケーションは LDAP に対応しているので、LDAP ディレクトリから人事管理データベースへ、またその逆方向へデータを自動的に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、その社員の電話番号への変更を LDAP ディレクトリと人事管理データ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、適切な処理として許容できます。ところが、この情報が古い場合 (たとえば、人事管理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を決めておきます。データ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

  この方法では、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ります。
- グループ内のユーザに読み取りと書き込みのアクセス権限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 (米国の場合は社会保障番号)、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、70 ページの「ディレクトリエントリのグループ化」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を決定するときに、複数のユーザ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。

iPlanet Delegated Administrator を使用すると、アカウントの管理業務を分割し、組織内のさまざまなロールを持つ個々のユーザに、アカウントの管理責任を委託できます。詳細は、iPlanet の販売代理店に問い合わせるか、http://www.iplanet.com にある iPlanet の Web サイトをご覧ください。

ディレクトリのアクセス制御の設定については、第 7 章の 127 ページの「安全なディレクトリの設計」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込め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、次の事項を決定する必要があります。

- 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) しないかぎり、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の記録を残すことができます。

  アクセス制御については、142 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、57 ページの第 4 章「ディレクトリツリーの設計」を参照してください。アクセス制御については、第 7 章の 127 ページの「安全なディレクトリの設計」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、サイトで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に与えなければならない場合があります。必ず社内の法務部に問い合わせて、ディレクトリの導入が、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導入方法については、第 7 章の 127 ページの「安全なディレクトリの設計」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きます。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。使い慣れたワードプロセッサでこの表を作成したり、表の内容を簡単に保存・検索できるようにスプレッドシートを使用することもできます。

次に、この簡単な表の例を示します。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。

データ名	所有者	マスターサーバ / 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
社員名	HR	People Soft	読み取り専用	(匿名)		
ユーザパスワード	IS	Directory US-1	読み取り / 書き込み	×	×	
自宅の電話番号	HR	People Soft	読み取り / 書き込み	×		×

データ名	所有者	マスターサーバ / 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
社員の所属場所	IS	Directory US-1	読み取り専用	(ログインが必要)	×	
会社の電話番号	施設	電話スイッチ	読み取り専用	(匿名)	×	×

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 / 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

  人事グループのメンバーは、ディレクトリ内の社員名を変更、追加、および削除できます。

- 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、ディレクトリ内の社員名を変更、追加、および削除できます。

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。

第 3 章

スキーマの設計

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、格納する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 (schema) 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、ディレクトリに格納するデータの一貫性を維持できます。

この章では、ディレクトリスキーマの内容と個々の要件に応じたスキーマを設計する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。

- スキーマ設計の概要
- iPlanet 標準スキーマ
- デフォルトスキーマへのデータの割り当て
- スキーマのカスタマイズ
- データの整合性の維持
- その他のスキーマ関連資料

スキーマの複製方法については、124 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。

スキーマ設計の概要

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

- スキーマの保守計画

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によってレビューされ、承認されています。

iPlanet 標準スキーマ

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、ディレクトリに格納されるデータの一貫性を維持します。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、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 に付属している定義済みのスキーマには、標準 LDAP スキーマと、サーバの機能をサポートするために追加された (iPlanet) アプリケーション固有のスキーマが含まれています。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、「スキーマのカスタマイズ」を参照してください。

次に、iPlanet 標準スキーマに含まれる形式、標準属性、およびオブジェクトクラスについて説明します。

スキーマの形式

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、Directory Server 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 LDAPv3 のスキーマと多少異なり、独自の専用オブジェクトクラスと属性を使用します。また、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 iPlanet で定義されている場合は、`X-ORIGIN` フィールドに `iPlanet 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 (objectlass $ 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。

標準属性

属性は、名前やファックス番号などの特定のデータ要素を保持します。Directory Server はデータを属性とデータのペア、つまり特定の情報に関連付けられた説明属性 (attribute) で表します。たとえば、commonName (cn) のようにユーザ名と標準属性をペアにして、データをディレクトリに格納できます。したがって、Babs Jensen という名前のユーザのエントリは、次の属性とデータのペアを持ちます。

```
cn: Babs Jensen
```

実際には、エントリ全体は一連の属性とデータのペアで表されます。たとえば、Babs Jensen のエントリは次のようになります。

```
dn: uid=bjensen, ou=people, dc=siroe,dc=com
objectClass: top
objectClass:person
objectClass: organizationalPerson
objectClass:inetOrgPerson
cn: Babs Jensen
sn: Jensen
givenName: Babs
givenName: Barbara
mail: bjensen@siroe.com
```

Babs のエントリでは、いくつかの属性に複数の値があります。属性 givenName は 2 回使用され、それぞれ固有の値が指定されています。この例にあるオブジェクトクラスについては、次の節「標準オブジェクトクラス」で説明します。

スキーマの各属性定義には、次の情報が含まれます。

- 一意の名前
- 属性のオブジェクト識別子 (object identifier) (OID)
- テキストによる属性の説明
- 属性構文の OID

- 属性が単一値と複数値のどちらであるか、そのディレクトリそのもので使用されるか、属性の作成元、属性に関連付けられた追加のマッチング規則

たとえば、cn 属性の定義はスキーマ内で次のように示されます。

```
attributetypes: ( 2.5.4.3 NAME 'cn' DESC 'commonName Standard
 Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

SYNTAX は属性値の構文の OID です。属性で使用できる構文は、Directory Server のバージョン 4.x から名前が変更され、拡張されています。次の表では、すべての構文の新しい名前の OID を示します。

表 3-1 属性の構文の定義

構文と OID	定義
Binary (以前は bin) 1.3.6.1.4.1.1466.115.121.1.5	属性値がバイナリ形式であることを示す
Boolean 1.3.6.1.4.1.1466.115.121.1.7	属性値が True と False のいずれかであることを示す
Country String 1.3.6.1.4.1.1466.115.121.1.11	属性値が印刷可能な 2 文字に制限されることを示す (fr など)
DN (以前は dn) 1.3.6.1.4.1.1466.115.121.1.12	属性値が DN (識別名) であることを示す
DirectoryString (以前は cis) 1.3.6.1.4.1.1466.115.121.1.15	属性値が大文字と小文字を区別しないことを示す
GeneralizedTime 1.3.6.1.4.1.1466.115.121.1.24	属性値が印刷可能な文字列として符号化されることを示す。タイムゾーンを指定する必要がある。GMT を使用することを強く推奨する
IA5String (以前は ces) 1.3.6.1.4.1.1466.115.121.1.26	属性値が大文字と小文字を区別することを示す
INTEGER (以前は int) 1.3.6.1.4.1.1466.115.121.1.27	有効な属性値が数字であることを示す
OctetString 1.3.6.1.4.1.1466.115.121.1.40	binary と同じ

表 3-1　属性の構文の定義 (続き)

構文と OID	定義
Postal Address 1.3.6.1.4.1.1466.115.121.1.41	属性値が次のように符号化されていることを示す。 *dstring* [$ *dstring*]* 各 *dstring* コンポーネントは DirectoryString 構文の値と同様に符号化される。*dstring* 内のバックスラッシュとドル記号は、区切り文字と間違えられることがないように、引用符で囲む。多くのサーバで、postal address は 最大 30 文字の 6 行に制限されている。たとえば、次のようにする `1234 Main St.$Anytown, TX 12345$USA`
TelephoneNumber (以前は `tel`) 1.3.6.1.4.1.1466.115.121.1.50	属性値が電話番号の形式であることを示す。国際形式の電話番号を使用することを推奨する
URI 1.3.6.1.4.1.250.1.57	属性値が `http://`、`https://`、`ftp`、`LDAP` などの文字列で始まる URL の形式であることを示す。URI は IA5String と同じである。RFC 2396 を参照

LDAPv3 のスキーマ形式については、『LDAPv3 Attribute Syntax Definitions document (RFC 2252)』を参照してください。

標準オブジェクトクラス

オブジェクトクラスは関連情報をグループ化するために使用します。通常、オブジェクトクラス (object class) は、個人や FAX 機器などの実際の対象を表します。オブジェクトクラスとその属性をディレクトリで使用するには、それらがスキーマ内で識別される必要があります。ディレクトリはデフォルトでオブジェクトクラスの標準リストを認識します。詳細は、『iPlanet Directory Server スキーマリファレンス』を参照してください。

各ディレクトリエントリは、1 つ以上のオブジェクトクラスに所属します。スキーマ内で識別されるオブジェクトクラスをエントリに配置することで、そのエントリにはある属性値のセットがあり、通常はそのセットよりも少ない別の属性値のセットがあることを Directory Server に知らせます。

オブジェクトクラスの定義には、次の情報が含まれます。

- 一意の名前
- オブジェクトを指定するオブジェクト識別子 (object identifier) (OID)
- 必須の属性のセット

- 許可された属性のセット

スキーマに示される標準オブジェクトクラスの例については、40 ページの「スキーマの形式」を参照してください。

iPlanet Directory Server のすべてのスキーマと同様に、オブジェクトクラスは直接 Directory Server で定義され、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。

デフォルトスキーマへのデータの割り当て

28 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既存の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、既存のデフォルトスキーマを表示する方法と、適切な既存の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。

既存の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、46 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

iPlanet Directory Server 5.1 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。

- Solaris 9 プラットフォームでは `/var/ds5/slapd-`*`serverID`*`/config/schema`
- Solaris 8 およびその他の UNIX プラットフォームでは `/usr/iplanet/servers/slapd-`*`serverID`*`/config/schema`
- Windows プラットフォームでは `C:\iPlanet\Servers\slapd-`*`serverID`*`\config\schema`

このディレクトリには、iPlanet 製品の共通スキーマがすべて入っています。LDAPv3 標準のユーザ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。

ディレクトリにある各オブジェクトクラスと属性については、『iPlanet Directory Server スキーマリファレンス』を参照してください。スキーマファイルとディレクトリ設定属性については、『iPlanet Directory Server 構成、コマンド、およびファイルのリファレンス』を参照してください。

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。

- データを表す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ディレクトリ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、ディレクトリ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

  最善の方法は、groups、people、organizations などの共通オブジェクトクラスを使用することです。

- 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、46 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第 2 章の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 3-2 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	Facilities	person	telephoneNumber

表では、社員名で個人を表しています。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。

電話番号も個人を表すものですが、person オブジェクトクラスに関連付けられたリストに該当する属性はありません。自宅の電話番号で考えると、これは組織の企業ネットワークにおける個人を表すものといえます。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 organizationPerson オブジェクトクラスから継承し、organizationPerson 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の中に、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 Console は、スキーマ定義の管理に役立ちます。詳細は、『iPlanet Directory Server 管理者ガイド』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
- 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。

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。

installDir/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

Directory Server が提供するオブジェクトクラスと属性は、ユーザ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 (object identifier) (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。独自のマッチング規則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。

作成するすべての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、siroe.com 社では、各カスタムスキーマ要素の前に `siroe` という接頭辞を追加しています。また、ディレクトリ内の siroe.com 社員を識別するために `siroePerson` 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戦略

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。

たとえば、属性 `siroeDateOfBirth`、`siroePreferredOS`、`siroeBuildingFloor`、および `siroeVicePresident` をサイトに作成するとします。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。`siroePerson` 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 `siroeDateOfBirth` と `siroePreferredOS` を許可するようにします。`siroePerson` の親は `inetOrgPerson` であるとします。`siroeOrganization` 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 `siroeBuildingFloor` と `siroeVicePresident` を許可するようにします。`siroeOrganization` 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: ( 2.16.840.1.17370.999.1.2.3 NAME 'siroePerson'  DESC
'Siroe Person Object Class' SUP inetorgPerson MAY (siroeDateOfBirth
 $ siroePreferredOS) )

objectclasses: ( 2.16.840.1.17370.999.1.2.4 NAME
 'siroeOrganization' DESC 'Organization Object Class' SUP
 organization MAY (siroeBuildingFloor $ siroeVicePresident) )
```

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: (2.16.840.1.17370.999.1.2.5 NAME 'siroeEntry' DESC
 'Standard Entry Object Class' SUP top AUXILIARY MAY
 (siroeDateOfBirth $ siroePreferredOS $ siroeBuildingFloor $
 siroeVicePresident) )
```

新しい `siroeEntry` 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 例にある新しいオブジェクトクラスの OID は、iPlanet OID 接頭辞に基づいています。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、47 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。

目的に合った、新しいオブジェクトクラスを定義する方法を選択してください。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 つのオブジェクトを使用する方が簡単です。

- 複数の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。この手間を有用と思う人もいれば、面倒だと思う人もいます。

- 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` 属性をユーザ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。

- 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、新しい属性とオブジェクトクラスを追加します。

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。対応する属性がデフォルトのディレクトリスキーマにない場合は、新しい属性を作成します。

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。ディレクトリに誕生日を格納する場合、iPlanet 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 `siroePerson` 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。ただし、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、その使用しない要素を削除してもかまいません。オブジェクトクラスの定義をスキーマから削除する前に、そのオブジェクトクラスを使用する各エントリを変更する必要があります。先に定義を削除すると、そのオブジェクトクラスを使用するエントリを変更できなくなる場合があります。不明なオブジェクトクラス値をエントリから削除しないかぎり、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

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の名前には、文字コードの昇順で並べたときに「99user.ldif」よりもあとになるものを使用すると、サーバに問題が発生する場合があります。

99user.ldif ファイルには X-ORIGIN 'user defined' という属性が含まれています。99zzz.ldif という名前のスキーマファイルを作成すると、次回に LDAP または Directory Server Console を使用してスキーマを更新したときに、X-ORIGIN 'user defined' を持つすべての属性が 99zzz.ldif に書き込まれます。ディレクトリがそれらの属性を 99zzz.ldif に書き込むのは、内部のスキーマ管理において、数字そして英字の順にもっとも後にくる英数字の名前をディレクトリが使用するためです。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。

カスタムススキーマファイルに名前を付けるときは、次の形式を使用します。

[00-99] *ユーザ名* .ldif

ディレクトリはこのようなスキーマファイルを英数字の順で読み込みます。したがって、数字が最初に読み込まれます。そのため、ディレクトリの定義済み標準スキーマより大きい数字のスキーマを使用する必要があります。たとえば、siroe.com 社で 60siroecorp.ldif という名前の新しいスキーマファイルを作成したとします。作成したスキーマファイルに文字コードの昇順で並べたときに標準スキーマファイルより前にくる名前を付けると、サーバがスキーマを読み込むときにエラーが発生することがあります。また、すべての標準属性とオブジェクトクラスが読み込まれるのは、カスタムスキーマ要素の読み込み後になります。

'user defined' は、スキーマが LDAP 経由で追加されたときにディレクトリが内部で使用するので、カスタムスキーマファイルの X-ORIGIN フィールドには 'user defined' を使用しないでください。'siroe.com Corporation defined' などのように、もっとわかりやすい記述を使用します。

カスタムスキーマファイルを作成したら、次のいずれかを実行できます。

- 作成したカスタムスキーマファイルをすべてのサーバに手動でコピーする。この場合は、各サーバを再起動する必要がある
- レプリケーションを使用して、この情報を各コンシューマに複製する

作成したカスタムスキーマファイルをすべてのサーバにコピーしなかった場合は、スキーマ情報は、LDAP または Directory Server Console を使用してサプライヤ上のスキーマを変更した場合にのみ、コンシューマに複製されます。

スキーマ定義がすでに存在していないコンシューマサーバに複製された場合、その定義は 99user.ldif ファイルに格納されます。ディレクトリはスキーマ定義が格納された場所を追跡することはありません。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、マスターサーバだけでスキーマを管理しているかぎり、問題はありません。

カスタムスキーマファイルを各サーバにコピーした場合は、スキーマファイルを変更したときに、変更したスキーマファイルを各サーバにコピーし直す必要があります。コピーし直さないと、変更がコンシューマ上の 99user.ldif ファイルにレプリケートおよび格納される可能性があります。変更が 99user.ldif ファイル内に格納されると管理がむずかしくなります。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。

スキーマの複製については、124 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。

最適なカスタムスキーマの作成

カスタムのスキーマ要素を作成するときは、次の点に留意します。

- LDAP または Directory Server Console を使用してスキーマを管理する場合は、複数のファイルでスキーマ要素が重複しないように、すべてのスキーマ定義を 99user.ldif に追加する。LDAP 経由で追加および更新されたスキーマ要素は、自動的に 99user.ldif ファイルに書き込まれる

注	たとえば、60siroecorp.ldif というカスタムスキーマファイルを定義し、これらのスキーマ要素を LDAP を使用して更新した場合、新しい定義はカスタムスキーマファイルにではなく 99user.ldif ファイルに書き込まれ、元のカスタムスキーマ定義が上書きされます。たとえば、60siroecorp.ldif への変更は 99user.ldif に格納されている定義で上書きされます。

- スキーマ要素を手動で 99user.ldif に追加する場合は、必ず 'user defined' の値を持つ X-ORIGIN を使用する。'user defined' 以外の値を使用すると、サーバは、99user.ldif ファイルからスキーマを読み込むときに、X-ORIGIN 用に指定したその値に加えて、定義の X-ORIGIN 部に 'user defined' 値を追加する。その結果、'user defined' ではない属性が Directory Server Console の読み取り専用セクションに表示され、Console を使用して 'user defined' 以外の X-ORIGIN を含むオブジェクトクラスを編集できなくなる

'user defined' の値を持つ X-ORIGIN を使用すると、99user.ldif ファイル内のスキーマ定義がディレクトリによってファイルから削除されることはありません。ディレクトリは 'user defined' という X-ORIGIN に基づいて 99user.ldif ファイルに残す要素を判別しているため、これらの定義を削除しません。

たとえば、次のようなスキーマエントリを 99user.ldif 内に手動で作成したとします。

```
attributetypes: ( siroeContact-oid NAME 'siroeContact' DESC
 'Siroe Corporate contact' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 X-ORIGIN 'Siroe defined')
```

ディレクトリがスキーマのエントリを読み込むと、次のようになります。

```
attributetypes: ( siroeContact-oid NAME 'siroeContact' DESC
 'Siroe Corporate contact' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 X-ORIGIN ('Siroe defined' 'user defined') )
```

- 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る。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る
- 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る。これにより、サーバ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る (サーバが最初に数字順、次にアルファベット順にスキーマを読み込むため)

データの整合性の維持

Directory Server 内のスキーマの一貫性を維持すると、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タイプと属性を選択し、常に同じものを使用する必要があります。一貫性のないスキーマオブジェクトを使用すると、ディレクトリツリー内の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、一貫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
- 一貫性のあるデータ形式を選択して適用する

次に、スキーマ内で一貫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。

スキーマ検査 (schema checking) はデフォルトで有効になっています。スキーマ検査は無効にしないことをお勧めします。スキーマ検査の有効と無効の切り替えについては、『iPlanet Directory Server 管理者ガイド』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`) と `surname` (`sn`) 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。

一貫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に適切な形式を選択して、ディレクトリツリー内で一貫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 iPlanet 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
```

レプリカされたスキーマでの一貫性の維持

レプリケート環境で iPlanet 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 Directory Server にわたってスキーマの一貫性を維持する必要があります。このレベルの一貫性を確保するための唯一の方法は、1 つのマスターサーバでのみスキーマの変更を行うことです。

ディレクトリのスキーマを変更すると、その変更は更新履歴ログに記録されます。複製時には、更新履歴ログ内の変更がスキャンされ、すべての変更が複製されます。レプリケート環境で一貫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のスキーマを変更しない

  コンシューマサーバのスキーマを変更すると、マスターサーバのスキーマよりも新しいスキーマとなります。したがって、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。

- 2 つのマスターサーバのスキーマを変更しない

  2 つのマスターサーバ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一貫性がないことを意味します。

- マルチマスターレプリケーション環境の場合も、必ず 1 つのマスターサーバでスキーマを変更する

スキーマレプリケーションについては、124 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

ディレクトリツリーの設計

ディレクトリツリー (directory tree) は、ディレクトリに格納されているデータを参照できるようにします。ディレクトリに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、および使用する複製のタイプによって、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を設計する手順の概要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
- ディレクトリツリーの設計例
- 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の設計は、ディレクトリデータの分散方法、レプリケート方法、あるいはアクセス制御方法など、ディレクトリの設計段階におけるさまざまな決定事項と密接に関係しています。導入前にディレクトリツリーの設計に多くの時間を割けば、ディレクトリを使い始めてからディレクトリツリーに不十分な点を見つけても、頭痛の種を減らすことができます。

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

- ディレクトリユーザ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、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。たとえば、各データベースでは、接尾辞レベルでデータを分割する必要があります。適切なディレクトリツリー構造がなければ、複数のサーバ間で希望どおりにデータを分散することができない場合があります。

また、使用するディレクトリツリー構造のタイプによって複製が制約を受けます。複製が機能するように、慎重にデータの分割を定義する必要があります。ディレクトリツリーの一部分だけを複製する場合は、設計段階からそのことを考慮に入れておく必要があります。また、分岐点でアクセス制御を使用したいのであれば、そのこともディレクトリツリーの設計段階で考慮しておきます。

ディレクトリツリーの設計

この節では、ディレクトリツリーの設計段階で決定する事柄について説明します。ディレクトリツリーの設計段階は、次の手順に分けられます。

- データを格納する接尾辞を選択する
- データエントリ間の階層構造の関係を決定する
- ディレクトリツリー階層内にあるエントリに名前を付ける

次に、ディレクトリツリーの設計段階について詳しく説明します。

接尾辞の選択

接尾辞 (suffix) は、ツリーのルートにあるエントリの名前です。接尾辞の下にディレクトリデータが格納されます。ディレクトリには複数の接尾辞を持たせることができます。一般的なルートポイントを持たない情報のディレクトリツリーが複数ある場合、複数の接尾辞を使用することもできます。

デフォルトでは、iPlanet Directory Server を標準的な構成で導入すると、データの格納用に 1 つの接尾辞が、構成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接尾辞が使用されます。標準のディレクトリの接尾辞については、『iPlanet Directory Server 管理者ガイド』を参照してください。

接尾辞の命名規則

ディレクトリ内のすべてのエントリは、ルート接尾辞 (root suffix) である共通のベースエントリの下に格納する必要があります。ディレクトリのルート接尾辞に名前を付けるときは、次のことを考慮してください。

- 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
- その接尾辞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、企業の DNS 名またはインターネットドメイン名に合わせてディレクトリの接尾辞を選択します。たとえば、企業が `siroe.com` というドメイン名を所有している場合は、次のようなディレクトリ接尾辞を使用します。

```
dc=siroe,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接尾辞を表します。

通常、ルート接尾辞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、ホスティングサービス事業者環境では、ルート接尾辞に使用する属性は次のものに限定することをお勧めします。

- `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`

  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。

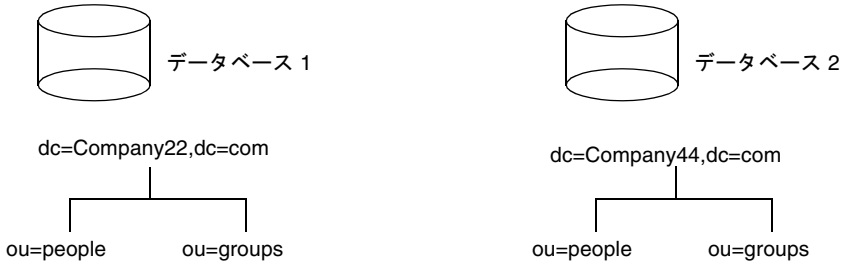
これらの属性が接尾辞に含まれていると、顧客のアプリケーションとの相互運用性が高まります。たとえば、ホスティングサービス事業者がこれらの属性を使用して、クライアントの Company22 に、次のようなルート接尾辞を作成する場合を考えてみます。

```
o=Company222,st=Washington,c=US
```

組織名のあとに国名コードを使用する指定方法は、X.500 の接尾辞の命名規則に従っています。

複数の接尾辞の命名

ディレクトリで使用する各接尾辞が、それぞれ固有のディレクトリツリーを示します。ディレクトリに複数のツリーを含めるには、いくつかの方法があります。1 つ目の方法は、複数のディレクトリツリーを作成し、各ディレクトリツリーを Directory Server によって提供される別々のデータベースに格納する方法です。たとえば、次のように Company22 と Company44 用に別々の接尾辞を作成し、それぞれを異なるデータベースに格納できます。



データベースは、資源の制限に応じて 1 つのサーバまたは複数のサーバに格納できます。

ディレクトリツリー構造の作成

ツリーの構造をフラットなものにするか階層型にするかを決定する必要があります。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、あとで複数のデータベース間にデータを分散したり、レプリケーション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。

ツリー構造を決定するには、次の手順と検討事項に従います。

- 61 ページの「ディレクトリの分岐点の作成」
- 62 ページの「分岐点の特定」
- 64 ページの「レプリケーションに関する検討事項」
- 66 ページの「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

階層を設計するときは問題の生じる名前の変更を避けるようにします。ネームスペースがフラットなほど、名前を変更する確率は低くなります。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。

次に、ディレクトリツリーの階層を設計するときのガイドラインを示します。

- 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る

  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。ディレクトリツリーを分岐させる部門は、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。

- 分岐点には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する

  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」を使用する)。

- 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 つ作成する

  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、すべてのマーケティングエントリをそのツリーに所属させます。

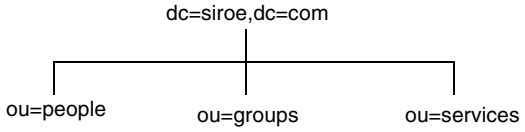
次に、会社とホスト環境に固有のガイドラインを示します。

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。

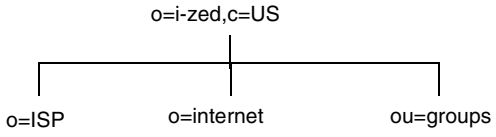
- `ou=people`
- `ou=groups`
- `ou=contracts`
- `ou=employees`
- `ou=services`

これらのオブジェクトを使用して構成されたディレクトリツリーは、次のようになります。



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接尾辞の下に含むツリーを作成します。たとえば、ISP の I-Zed 社では次のようにディレクトリを分岐させます。

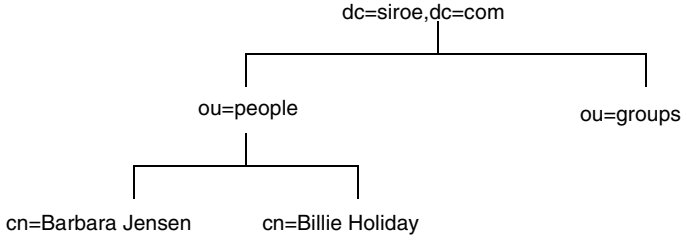


分岐点の特定

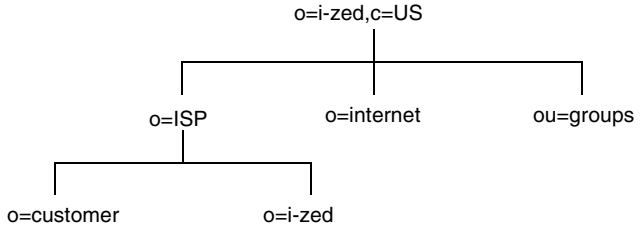
ディレクトリツリーをどのように分岐させるかを決めるには、その分岐点を特定するために使用する属性を決定することが必要になります。DN は、属性とデータのペアで構成される一意の文字列です。たとえば、siroe.com 社の社員 Barbara Jensen 用のエントリの DN は次のようになります。

```
cn=Barbara Jensen,ou=people,dc=siroe,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。たとえば、siroe.com 社という企業のディレクトリツリーは次のようになります。



インターネットのホストである I-Zed 社のディレクトリツリーは、次のようになります。



ルート接尾辞のエントリ `o=i-zed,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 I-Zed の内部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、次のような事柄を検討する必要があります。

- 一貫性のあるものにする

  ディレクトリツリー全体で識別名 (distinguished name) (DN) の形式が統一されていないと、いくつかの LDAP クライアント (LDAP client) アプリケーションが混乱する可能性があります。つまり、ディレクトリツリーのある部分で `l` が `o` 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してください。

- よく使用されている従来型の属性だけを使用する (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する。また、58 ページの「接尾辞の命名規則」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る

表 4-1　従来からある DN 分岐点の属性 (続き)

属性名	定義
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する。組織単位は、一般にすぐ上の組織に属する
st	州あるいは県の名前
l	地域 (都市、地方、オフィス、施設名など)
dc	58 ページの「接尾辞の命名規則」で説明されているドメインのコンポーネント

注　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。

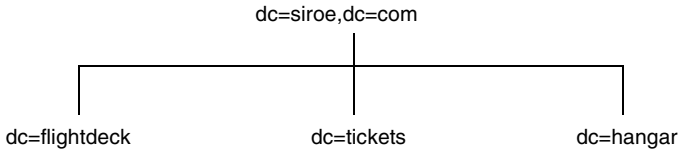
ただし、エントリ自体に格納された属性とデータのペアに基づいてエントリを検索することは可能です。したがって、あるエントリの識別名が cn=Babs Jensen,ou=People,dc=siroe,dc=com の場合も、dc=siroe を対象とした検索は、そのエントリ内に属性として dc: sun を明示的に置かないかぎり、そのエントリとはマッチしません。

レプリケーションに関する検討事項

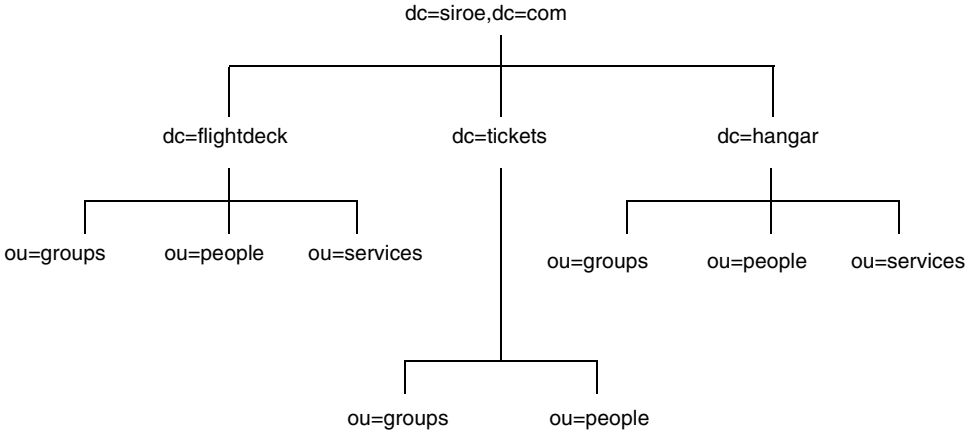
ディレクトリツリーの設計時に、レプリケーションするエントリを検討します。エントリセットをレプリケーションする場合は、サブツリーの頂点で識別名 (distinguished name) (DN) を指定し、その下にあるエントリをすべてレプリケートするのが自然な方法です。また、このサブツリーは、ディレクトリデータの一部を含むディレクトリパーティションである、データベースに対応します。

たとえば、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、複製を使用して別の 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。

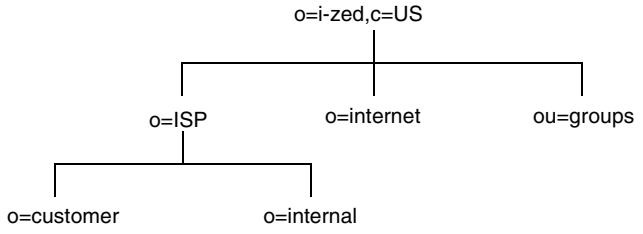
たとえば、siroe.com 社に flightdeck.siroe.com、tickets.siroe.com、および hanger.siroe.com という 3 つのプライマリネットワークがあるとします。これらのネットワークでは、最初にディレクトリツリーを次のように分岐させています。



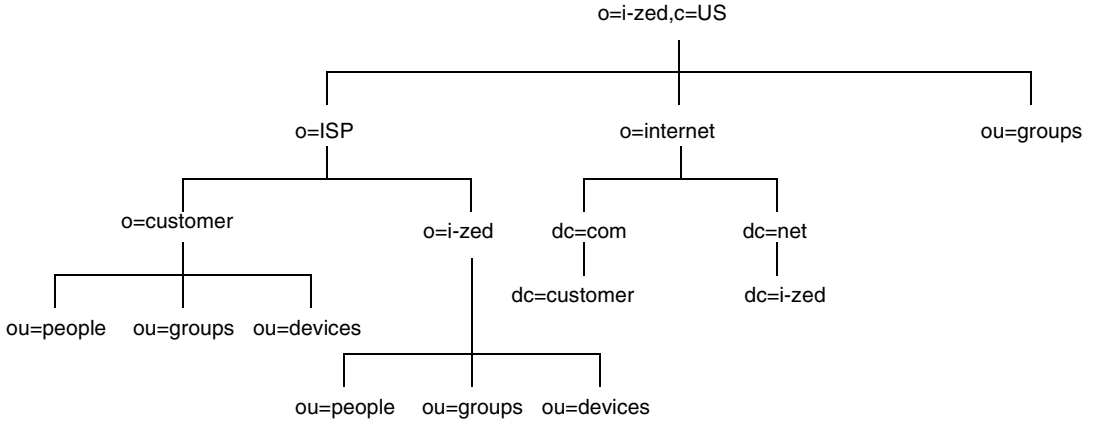
ツリーの最初の構造を作成したあと、ネットワークをさらに次のように分岐させています。



別の例として、ISP の i-zed.com 社は次のようにディレクトリを分岐させています。



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次のように分岐させています。



siroe.com 社と ISP の i-zed.com 社はどちらも、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。複製の場合と同様に、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ができます。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ができます。ACI フィルタを適用するターゲット (target) 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。

エントリの命名

ディレクトリツリー階層を設計したあとは、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、RDN (相対識別名 (Relative distinguished name)) を形成して作成します。RDN とは、一番左の DN 属性値のことです。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ります。

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。

- ユーザエントリの命名
- 組織エントリの命名
- その他のエントリの命名

ユーザエントリの命名

ユーザエントリの名前、DN は一意である必要があります。従来、識別名ではそのユーザ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=siroe,dc=com
```

このエントリと関連付けられているユーザを識別するのは簡単ですが、同じ名前のユーザが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 (name collisions) 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=siroe,dc=com
```

ただし、大きなディレクトリの場合は、扱いにくい共通名となり、管理が難しくなります。

より良い方法は、cn 以外の属性でユーザ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@siroe.com, dc=siroe,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 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。

ユーザエントリの 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、必ず一意で永続的な値を使用します。また、ユーザエントリの RDN は読みやすいものにします。識別名に基づいてディレクトリエントリを変更する場合など、DN を認識しやすくするとディレクトリのタスクが 簡単になるものがあります。つまり、uid=bjensen, dc=siroe,dc=com の方が uid=b12r56A, dc=siroe,dc=com より適切です。また、ディレクトリのクライアントアプリケーションの中には、uid 属性と cn 属性に人間が読める名前を使用することを前提にしているものもあります。

ホストされる環境でのユーザエントリに関する検討事項

ユーザ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。

DIT 内へのユーザエントリの配置

次に、ディレクトリツリーにユーザエントリを配置する場合のガイドラインを示します。

- 企業内のユーザのエントリは、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

組織エントリの命名

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=Company22+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含ませる必要があります。

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

その他のエントリの命名

地域、州、国、デバイス、サーバ、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=siroe,dc=com
```

ただし、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

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。グループとロールによってエントリを柔軟に関連付けられるようになり、サービスクラスによってディレクトリの分岐内で共有されるデータを簡単に管理できるようになります。

グループ化メカニズムについては、次の節で説明します。

- 静的グループと動的グループ
- 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
- サービスクラス

静的グループと動的グループ

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。

- 静的グループは、そのメンバーエントリの名前を明示的に指定する。静的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` の属性値として含む。静的グループは、ディレクトリ管理者のグループなど少人数のグループに適している

  静的グループは、メンバーが何千人といるようなグループには適しません。性能が著しく低下するため、メンバー数が 20,000 を超える静的グループを作成することはお勧めできません。これ以上のサイズのグループには、動的グループまたはロールを使用することをお勧めします。メンバー数が 20,000 を超えるグループを定義するために静的グループを使用する必要がある場合は、1 つの大きな静的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。

- 動的グループはフィルタを指定し、一致するすべてのエントリがグループのメンバーとなる。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に動的である。動的グループの定義エントリは `groupOfURLs` オブジェクトクラスに属し、`memberURL` 属性の LDAP URL 値として表現される 1 つまたは複数のフィルタを含む

両方のタイプのグループで、ディレクトリ内のあらゆるメンバーを特定できます。グループ定義自体を `ou=Groups` 分岐の下に配置することをお勧めします。たとえば、バインド資格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 ACI (Access Control Instruction) を定義するときに、これによって検索が簡単になります。

グループの利点は、すべてのメンバーを簡単に検索できることです。静的グループは簡単に一覧表示でき、動的グループのフィルタは簡単に評価できます。グループの欠点は任意のエントリをメンバーにできることです。このため、あるエントリがメンバーとなっているすべてのグループの名前を挙げることは、非常に困難です。

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

ロールはエントリをグループ化する新しいメカニズムで、エントリがメンバーとなっているすべてのロールを自動的に特定します。ディレクトリ内のエントリを検索すると、そのエントリが属しているロールがすぐにわかります。これによって、グループのメカニズムの主な欠点が解消されます。

- 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、静的グループと同じである。静的なロール定義エントリは、効果の範囲だけを定義し、その範囲はその親エントリの分岐全体である。そのロールのメンバーは分岐内のエントリであり、その `nsRoleDN` 属性でロール定義エントリの DN を指定している
- フィルタを適用したロールは動的グループと非常に似ていて、ロールのメンバーを決めるフィルタを定義する。すべてのロールと同様に、フィルタの範囲はフィルタを適用したロール定義エントリの場所によって定義される
- 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る。つまり、あるエントリが入れ子のロール定義のリストにさ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

Directory Server がすべての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は非常に簡単に使用できます。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。`nsRole` が仮想属性であるのは、実行中にサーバによって生成され、実際にはディレクトリに格納されないためです。

`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並べることができます。したがって、指定したエントリが特定のロールに属しているかどうかを簡単に判断できます。

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機能の一部が重複しているため、あいまいさを招く可能性があります。エントリをグループ化するどちらの方法にも、利点と欠点があります。とはいえ、新しいロールメカニズムは、頻繁に必要とされる機能を最大限に引き出せるように設計されています。

グループ化メカニズムを使用する主な理由は次の 2 つです。

- クライアントがグループまたはロールのすべてのメンバーを検索する必要がある

  グループがもっとも効率良く機能するのはこのときです。nsRole は仮想属性であり、フィルタでは使用できないため、ロールのすべてのメンバーを検索するのは困難です。ただし、クライアントは分岐内のすべてのエントリを検索し、nsRole 属性の値を読み取って、すべてのロールメンバーを検索することは可能です。

- エントリが特定のグループまたはロールのメンバーであるかどうかをクライアントが知る必要がある

  本来、これはコンピュータに大きな負荷がかかるタスクです。さらに、グループをもとにした実装は非常に複雑です。ロールはこの問題を解決するために開発されました。ロール定義の構造と範囲が決まっていれば、サーバはクライアントよりもずっと効率よくメンバーを算出できます。したがって、クライアントがあるタイプのロールのメンバーを検索するときは、生成された nsRole 属性の値を読み取るだけで済みます。

必要に応じて、以下の設計をお勧めします。

- メンバー数を算出するだけの場合は、静的グループを使用する。クライアントは静的グループ定義を検索し、すべてのメンバーの DN のリストを簡単に取得できる
- ACI でのバインド規則の指定など、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、動的グループを使用する。フィルタを適用したロールはフィルタを適用したグループとほとんど同じであるが、nsRole 仮想属性を生成するロールメカニズムが始動する。クライアントが nsRole 値を必要としない場合は、動的グループだけを定義すると計算のオーバーヘッドを回避できる
- クライアントが特定のエントリのすべてのメンバー情報を必要とする場合は、ロールを使用する。サーバ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済む。さらに、この属性にすべてのタイプのロールが現れ、クライアントはすべてのロールを均等に処理できる
- エントリのセットがいくつかあり、特定のセットのメンバーを並べること、および特定のエントリがメンバーとなっているすべてのセットを検索することが必要である場合は、ロールだけを使用する。全般的に、グループよりもロールの方が、両方を効率よく、より簡単なクライアントで実行できる

また、ツリー階層を使用して、グループと同等のエントリのセットを作成することもできます。これによって、ロールのメンバーを並べるのが簡単になり、ロールに基づく設計の効率を最大限に引き出すことができます。

グループ化メカニズムは、サーバ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。要件に合うメカニズムを知り、効率よく使用してください。最後に、管理者があとでポリシーの一貫性を維持できるように、選択した設計を文書化しておくことを検討してください。

サービスクラス

サービスクラス (Class of Service) (CoS: Class of Service) メカニズムを使用すると、アプリケーションに見えない方法で、エントリ間で属性を共有できます。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。

たとえば、ディレクトリには `facsimileTelephoneNumber` 属性の値が等しい何千ものエントリを格納できます。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ました。これは管理者にとって大きな負担であり、全てのエントリが更新されないという危険がありました。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。さらに、実際に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。また、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。

CoS メカニズムでは、2 つのタイプのヘルパーエントリが使用されます。

- CoS 定義エントリは、生成される属性とその値の指定方法を示す。定義エントリの場所によって、CoS 偏差の範囲が決まる。定義エントリの親の分岐内のすべてのエントリが、CoS 定義のターゲットエントリと呼ばれる

  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には、CoS 属性に生成される値が含まれる。値の異なる複数のテンプレートがある可能性がある。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テンプレートを選択する

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。

- ポインタ CoS はもっとも単純で、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 が決まる。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 属性値が設定される
- 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る。定義エントリは、ターゲットエントリにある指示子属性の名前を指定する。この属性には必ず DN が含まれ、その値によって、特定のターゲットに使用されるテンプレートが決まる

  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。

  間接 CoS は使いすぎないようにしてください。ディレクトリツリーの任意の場所にある任意のエントリをテンプレートにすることができるため、アクセスの制御が非常に困難になります。また、特に性能が重視される場所では、間接 CoS の使用は避けてください。
- クラシック CoS はポインタ CoS と間接 CoS の動作を組み合わせたもので、CoS 定義によって、テンプレートのベース DN と指示子属性の名前の両方が指定される。ターゲットエントリの指示子属性の値は、以下に示すようにテンプレートエントリの DN を構築するために使用する

  `cn=`*specifierValue,baseDN*

  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性能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。テンプレートを定義エントリと同じ場所に格納し、意味のある名前を付けて、CoS メカニズムを簡単に管理できるようにすることをお勧めします。

生成される CoS 属性には、複数のテンプレートにより複数の値を設定できる場合があります。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、選択し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、『iPlanet Directory Server 管理者ガイド』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 (role-based attributes) を指定できます。エントリは関連付けられたサービスクラス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、iPlanet Directory Server では、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、クライアントアプリケーション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、サーバの性能が低下します。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。

最後に、必要以上に CoS 定義を変更しないようにしてください。サーバは 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。キャッシュに保存することによって、生成される属性エントリへの読み取りアクセスが速くなります。CoS 情報が変更されると、サーバ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、読み取り処理では新たに変更された情報ではなく、古いキャッシュ情報にアクセスする可能性があります。

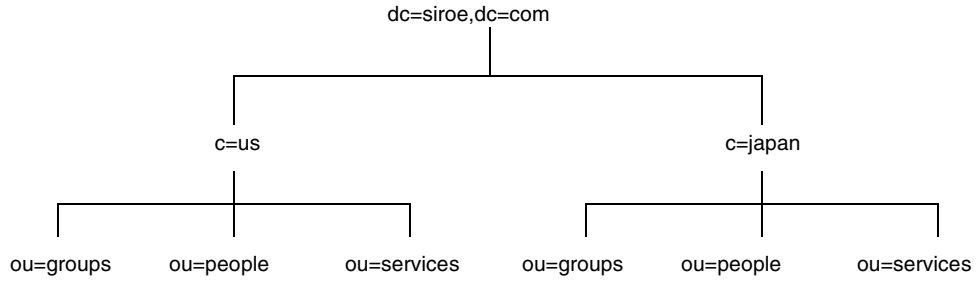
ディレクトリツリーの設計例

次に、フラットな階層をサポートするディレクトリツリーの設計例と、より複雑な階層を含む設計例を示します。

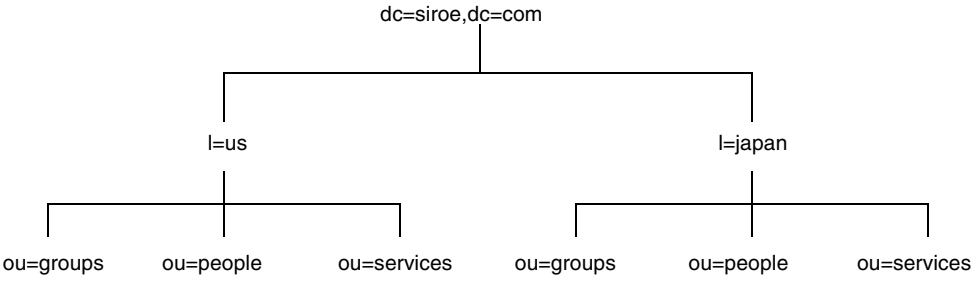
国際企業向けのディレクトリツリー

国際企業の要件に対応するには、ディレクトリツリーのルートをインターネットのドメイン名に設定し、そのルートポイントのすぐ下で、企業が事業を展開している国ごとにツリーを分岐させます。58 ページの「接尾辞の命名規則」では、国名指示子をディレクトリツリーのルートとして設定しないように勧めています。これは、企業が国際的な組織である場合には、特に当てはまります。

LDAP は DN 内の属性の順序については何の規制も設けていないため、次のように `c` (`countryName`) 属性を使用して各国の分岐を表すことができます。



ただし、この方法ではスタイルに統一感がないと感じる管理者もいるので、代わりに `l` (`locality`) 属性を使用して、各国を次のように表すこともできます。



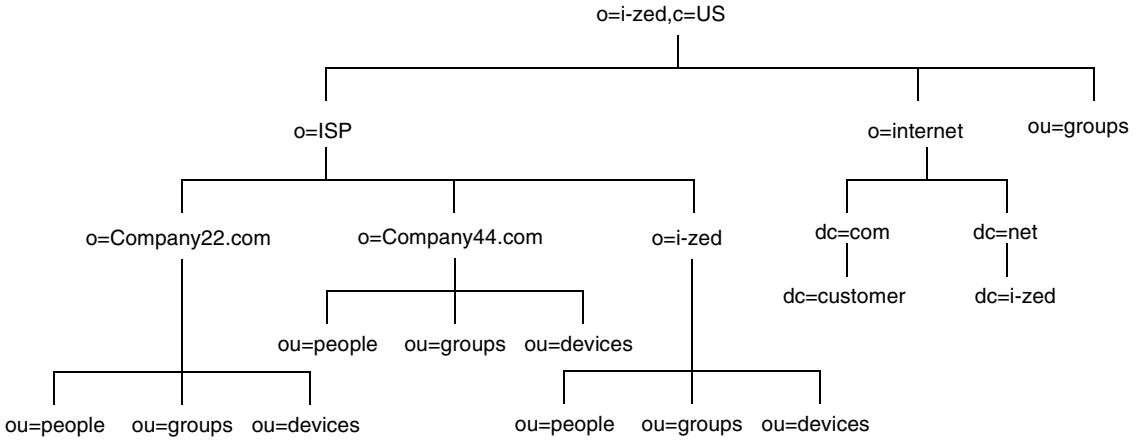
ISP 向けのディレクトリツリー

IPS (インターネットサービスプロバイダ) は、ディレクトリを通じて複数の企業をサポートする可能性があります。ISP の場合は、各顧客を個別の企業とみなし、ディレクトリツリーを設計します。セキュリティ上の理由から、それぞれが一意の接尾辞を持つ個別のディレクトリツリーをアカウントごとに提供し、各ディレクトリツリーに適した個別のセキュリティポリシーを設定します。

各顧客に別々のデータベースを割り当て、別々のサーバにこれらのデータベースを格納することもできます。各ディレクトリツリーを専用のデータベース内に置けば、ほかの顧客に影響を与えずに各ディレクトリツリーにあるデータをバックアップしたり復元したりできます。

また、データを分散して配置することにより、ディスクの競合に起因する性能上の問題を減らし、ディスク障害によって影響を受けるアカウントの数を減らすこともできます。

たとえば、ISP の I-Zed 社用のディレクトリツリーは次のようになります。



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は、次のリンクを参照してください。

- RFC 2247: Using Domains in LDAP/X.500 Distinguished Names (LDAP/X.500 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

ディレクトリトポロジの設計

第 4 章「ディレクトリツリーの設計」では、ディレクトリでのエントリの格納方法について説明しました。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 (topology) 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、ディレクトリのトポロジの計画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。

- トポロジの概要
- データの分散
- 知識参照について
- インデックスを使用したデータベース性能の向上

トポロジの概要

第 4 章「ディレクトリツリーの設計」で設計したディレクトリツリーが複数の物理的な Directory Server に分散されている、分散型のディレクトリを構成するように、iPlanet Directory Server の配置を設計できます。ディレクトリを複数のサーバに分割する手法は、次のような機能強化を実現するのに役立ちます。

- ディレクトリを使用するアプリケーションに最大限の性能を提供する
- 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

データベースは、複製、バックアップ、データの復元などの作業に使用する基本単位です。1 つのディレクトリを管理可能な複数のチャンクに分割し、それらのチャンクを異なるデータベースに割り当てることができます。これらのデータベースは多数のサーバに分散できるので、各サーバの作業負荷が軽減します。1 つのサーバに複数のデータベースを格納できます。たとえば、1 つのサーバに異なる 3 つのデータベースを格納できます。

ディレクトリツリーを複数のデータベースに分割した場合、各データベースには接尾辞 (suffix) と呼ばれるディレクトリツリーの一部分が格納されます。たとえば、ディレクトリツリーの `ou=people,dc=siroe,dc=com` という接尾辞、あるいは分岐にあるエントリを 1 つのデータベースに格納できます。

ディレクトリを複数のサーバに分割すると、各サーバ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 サーバに割り当てるドメインネームサービス (DNS) に似ています。DNS と同様に、ディレクトリのネームスペースを複数のサーバに分散し、クライアント側からは単一のディレクトリツリーを持つ 1 つのディレクトリとして管理させることができます。

iPlanet Directory Server は、別々のデータベースに格納されているディレクトリのデータをリンクするメカニズムである知識参照も提供します。Directory Server には、レフェラルと連鎖という 2 つのタイプの知識参照があります。

以降の節では、データベースと知識参照について説明し、2 つのタイプの知識参照の違いと、データベースの性能を向上させるためのインデックスの設計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、社内の各サーバ上にディレクトリのエントリを物理的に格納することなく、ディレクトリを複数のサーバにわたって拡張できます。そのため、分散型ディレクトリは、1 つのサーバ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。ユーザとアプリケーションからは、ディレクトリへの問い合わせに応答する単一のディレクトリがあるようにしか見えません。

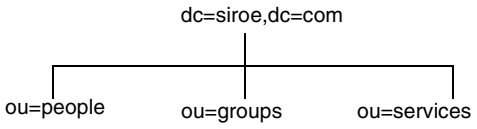
次に、データ分散のしくみについて詳しく説明します。

- 81 ページの「複数のデータベースの使用について」
- 82 ページの「接尾辞について」

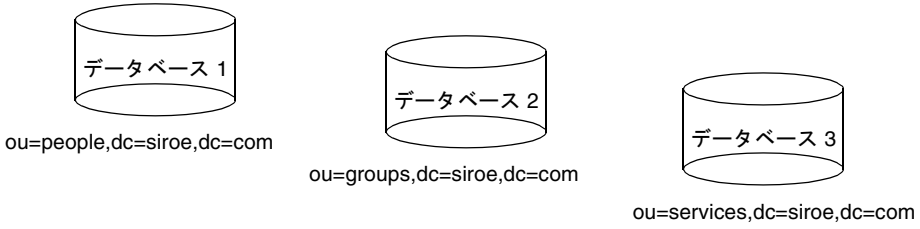
複数のデータベースの使用について

iPlanet 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 (LDBM database) はディスクベースの高性能データベースです。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。

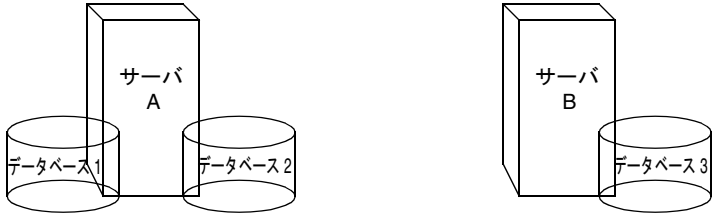
ディレクトリツリーの異なる部分を別々のデータベースに格納できます。たとえば、次のようなディレクトリツリーがあるとします。



ツリーにある 3 つの接尾辞のデータを、次のように 3 つの異なるデータベースに格納できます。



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に分散できます。たとえば、ディレクトリツリーの 3 つの接尾辞を含めるために作成した 3 つのデータベースを、次のように 2 つのサーバに格納できます。



サーバ A には 1 番目と 2 番目のデータベース、サーバ B には 3 番目のデータベースが含まれます。

データベースを複数のサーバに分散すると、各サーバ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。

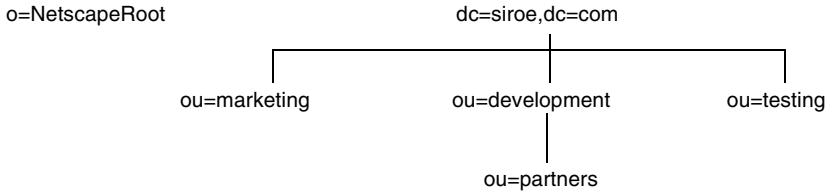
さらに、iPlanet Directory Server ではデータベースを動的に追加できるので、ディレクトリにデータベースが必要になったときに、ディレクトリ全体をオフラインにしなくても新しいデータベースを追加できます。

接尾辞について

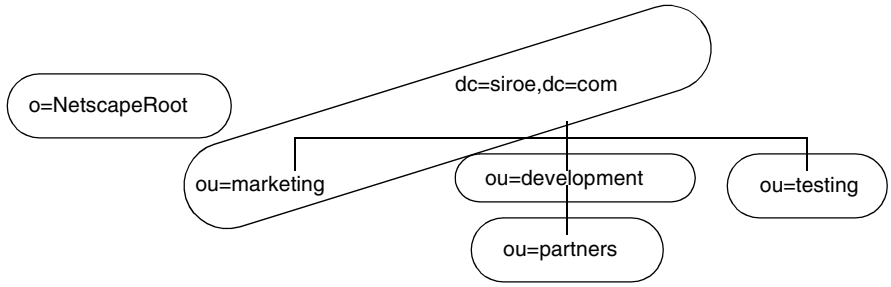
各データベースには Directory Server の接尾辞にあるデータが格納されます。ルート接尾辞とサブ接尾辞の両方を作成して、ディレクトリツリーの内容を編成できます。ルート接尾辞 (root suffix) はツリーの最上位にあるエントリです。この接尾辞は、ディレクトリツリーのルートか、または Directory Server 用に設計したもっと大きなツリーの一部を形成します。

サブ接尾辞 (sub suffix) はルート接尾辞の下にある分岐です。ルート接尾辞とサブ接尾辞のデータはデータベースに格納されます。

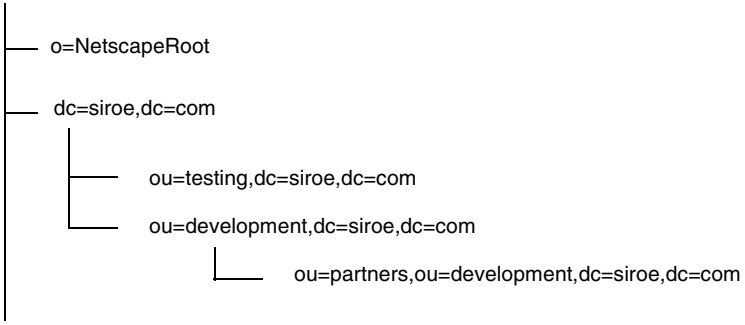
たとえば、ディレクトリデータの分散を表すために接尾辞を作成したいとします。siroe.com 社のディレクトリツリーは次のように表されます。



siroe.com 社は、ディレクトリツリーを、次のように 5 つの異なるデータベースに分割するとします。

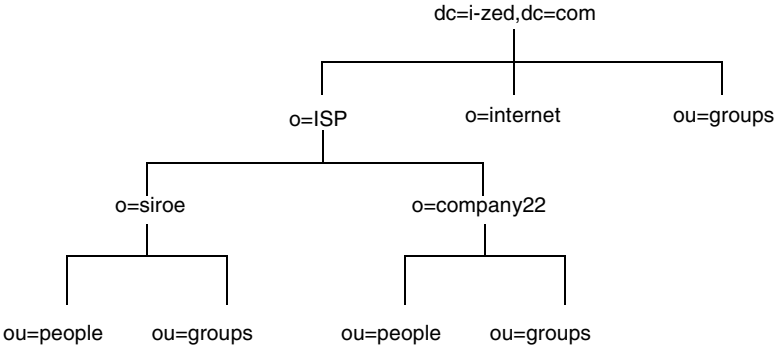


この結果、接尾辞には次のようなエントリが含まれます。



`o=NetscapeRoot` 接尾辞と `dc=siroe,dc=com` 接尾辞は両方ともルート接尾辞です。ほかの接尾辞である `ou=testing,dc=siroe,dc=com`、`ou=development,dc=siroe,dc=com`、および `ou=partners,ou=development,odc=siroe,dc=com` はすべて、`dc=siroe,dc=com` ルート接尾辞のサブ接尾辞です。ルート接尾辞 `dc=siroe,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。

ディレクトリに複数のルート接尾辞が含まれることもあります。たとえば、I-Zed 社という ISP が、siroe.com と company22.com のそれぞれに、複数の Web サイトをホストするとします。この ISP は、2 つのルート接尾辞を作成します。1 つは `dc=siroe,dc=com` 命名コンテクストに対応し、もう 1 つは `dc=company22,dc=com` 命名コンテクストに対応します。ディレクトリツリーは次のように表されます。



dc=i-zed,dc=com エントリはルート接尾辞を表します。各ホスト ISP のエントリもルート接尾辞 (o=siroe と o=company22) を表します。ou=people 分岐と ou=groups 分岐は、各ルート接尾辞の下にあるサブ接尾辞です。

知識参照について

データを複数のデータベースに分散したあとは、分散したデータ間の関係を定義する必要があります。そのためには、別々のデータベースに保持されているディレクトリ情報へのポインタである知識参照を使用します。iPlanet 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、次のタイプの知識参照があります。

- レフェラル

  サーバは、要求を完了するには別のサーバ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、これら 2 つのタイプの知識参照をさらに詳しく比較して説明します。

レフェラルの使い方

レフェラル (referral) はサーバが返す情報であり、操作要求を完了するために接続する必要があるサーバをクライアントアプリケーションに指示します。この転送メカニズムは、クライアントアプリケーションがローカルサーバに存在しないディレクトリエントリを要求したときに起動されます。

Directory Server は、次の 2 つのタイプのレフェラルをサポートしています。

- デフォルトレフェラル

  クライアントアプリケーションが提示した DN に対応する接尾辞がサーバにない場合、ディレクトリはデフォルトレフェラルを返します。デフォルトレフェラルはサーバの設定ファイルに格納されています。Directory Server のデフォルトレフェラルを設定することも、各データベースに個別のデフォルトレフェラルを設定することもできます。

  各データベースに設定したデフォルトレフェラルは、接尾辞設定情報を通じて実行されます。データベースの接尾辞が無効な場合は、その接尾辞に対して発行されたクライアント要求にデフォルトレフェラルを返すようにディレクトリを設定できます。接尾辞については、82 ページの「接尾辞について」を参照してください。接尾辞の設定方法については、『iPlanet Directory Server 管理者ガイド』を参照してください。

- スマートレフェラル

  スマートレフェラルは、ディレクトリ自体のエントリに格納されています。このレフェラルは、このスマートレフェラルが格納されているエントリの DN とマッチする DN を持つサブツリーに関する情報を保有している Directory Server をポイントします。

レフェラルはすべて、LDAP の URL (Uniform Resource Locator) 形式で返されます。次に、LDAP レフェラルの構造と、Directory Server がサポートする 2 つのタイプのレフェラルについて説明します。

LDAP レフェラルの構造

LDAP レフェラルには 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。

- 接続先サーバのホスト名
- サーバのポート番号
- 検索操作の場合はベース DN (base 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=siroe,dc=com` 内で検索するとします。レフェラルは、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.siroe.com:389/ou=people,l=europe,dc=siroe,dc=com

レフェラルは、ポート 389 のホスト europe.siroe.com に接続し、ou=people,l=europe,dc=siroe,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。

レフェラルがどのように処理されるかは、使用している LDAP クライアントアプリケーションによって決まります。転送されたサーバ上で操作を自動的に再試行するクライアントアプリケーションもあれば、単にレフェラル情報をユーザに返すだけのクライアントアプリケーションもあります。コマンド行ユーティリティなど、iPlanet 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、自動的にレフェラルを実行します。サーバへのアクセスには、最初のディレクトリ要求で提供するバインド資格が使用されます。

ほとんどのクライアントアプリケーションは、レフェラルの制限数、あるいはホップ数だけレフェラルを実行します。実行するレ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低減でき、また循環レ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。

デフォルトレフェラルについて

接続したサーバまたはデータベースに要求したデータがない場合は、デフォルトレフェラルがクライアントに返されます。

Directory Server は、要求されたディレクトリオブジェクトの DN とローカルサーバでサポートされているディレクトリ接尾辞を比較して、デフォルトレ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接尾辞がマッチしない場合はデフォルトレ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=siroe,dc=com

ところが、サーバは dc=europe,dc=siroe,dc=com 接尾辞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=siroe,dc=com 接尾辞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を示すレフェラルをクライアントに返します。デフォルトレフェラルを受け取ったクライアントは適切なサーバに接続して、元の要求を再送信します。

デフォルトレフェラルは、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 をポイントするように設定します。サーバのデフォルトレフェラルは、nsslapd-referral 属性で設定します。ディレクトリインストール時の各データベースのデフォルトレフェラルは、構成内のデータベースエントリの nsslapd-referral 属性で設定されます。これらの属性値は dse.ldif ファイルに格納されます。

デフォルトレフェラルの設定方法については、『iPlanet Directory Server 管理者ガイド』を参照してください。

スマートレフェラル

Directory Server では、スマートレフェラルを使用するようにディレクトリを設定することもできます。スマートレフェラルを使用すると、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。

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

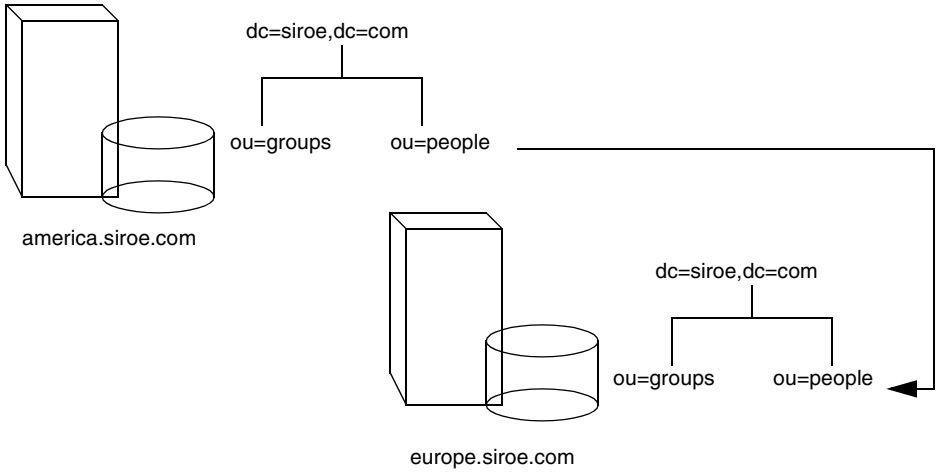
デフォルトレフェラルとは異なり、スマートレフェラルはディレクトリ自体に格納されます。スマートレフェラルの設定方法と管理については、『iPlanet Directory Server 管理者ガイド』を参照してください。

たとえば、siroe.com 社の米国支社のディレクトリに、`ou=people,dc=siroe,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レフェラルを指定することで、この分岐への要求をすべて siroe.com 社のヨーロッパ支社の `ou=people` 分岐に転送できます。このスマートレフェラルは、次のようになります。

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

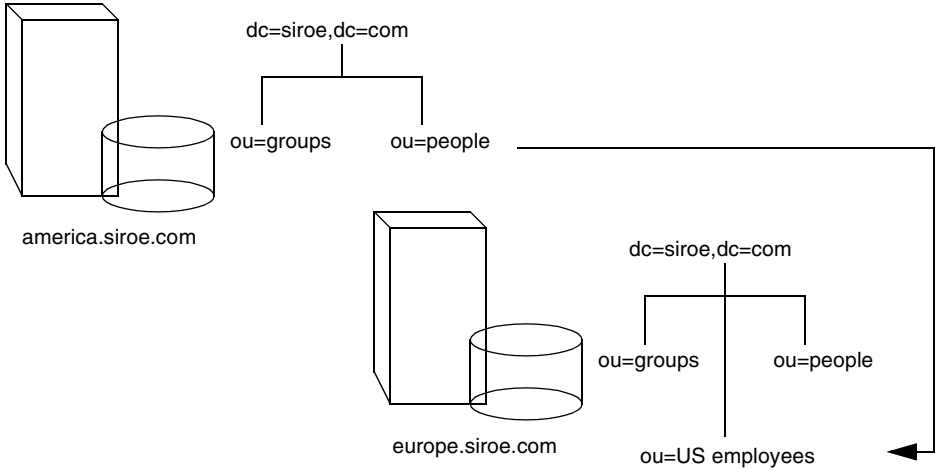
米国支店のディレクトリにある people 分岐への要求はすべて、ヨーロッパのディレクトリに転送されます。このスマートレフェラルを図に示すと、次のようになります。



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に問い合わせを転送できます。たとえば、siroe.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 siroe.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レフェラルを返します。

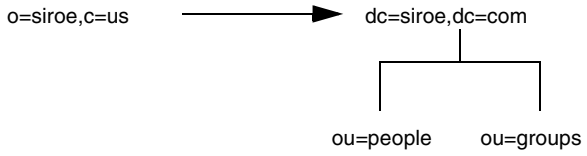
```
ldap://europe.siroe.com:389/ou=US employees,dc=siroe,dc=com
```

次の図に、このレフェラルの動作を示します。



同じサーバ上に複数の接尾辞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問い合わせを転送できます。ローカルマシン上の `o=siroe,c=us` への問い合わせをすべて `dc=siroe,dc=com` に転送する場合は、`o=siroe,c=us` エントリに次のスマートレフェラルを設定します。

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



この LDAP URL の 3 番目のスラッシュは、URL が同じ Directory Server をポイントしていることを示しています。

注	あるネームスペースから別のネームスペースへのレフェラルの作成は、レフェラルの識別名に基づいた検索を実行するクライアント以外では使用できません。`ou=people,o=siroe,c=US` の下の検索など、ほかの操作は正しく実行されません。

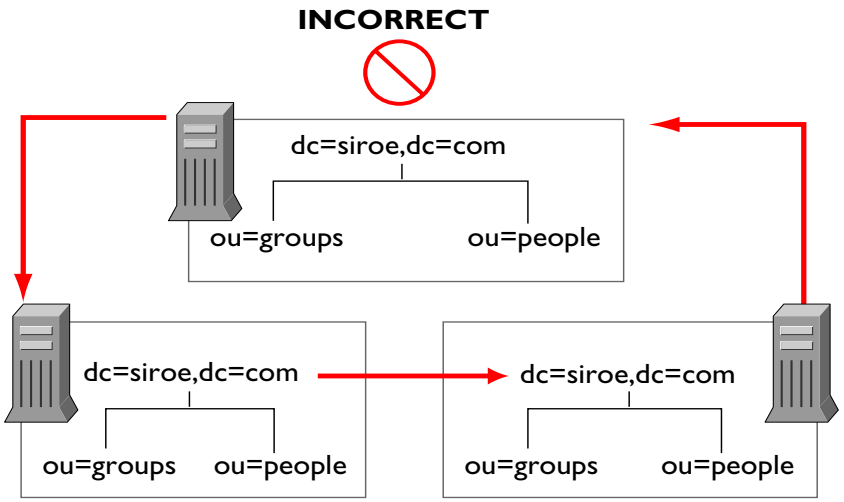
LDAP URL と、スマート URL を iPlanet Directory Server のエントリに含める方法については、『iPlanet Directory Server 管理者ガイド』を参照してください。

スマートレフェラルを設計する際のヒント

スマートレフェラルは簡単に使用できますが、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

  複雑に交錯したレフェラルを使用してディレクトリを運用すると、管理が難しくなります。また、スマートレフェラルを使いすぎると、循環レ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。例えば、レフェラルがある LDAP URL をポイントし、その LDAP URL が別の LDAP URL をポイントするといったことが、連鎖のどこかで最後に元のサーバに戻ってしまうまで続くという状況です。次の図に、循環レフェラルパターンを示します。



- 主要な分岐点で転送する

  ディレクトリツリーの接尾辞レベルで転送を処理するように、レフェラルの使用を制限します。スマートレ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や DN に転送できます。そのため、スマートレ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。レフェラルの使用をディレクトリツリーの接尾辞または主要な分岐点に限定することで、管理するレフェラル数が減少し、ディレクトリの管理に伴う負荷を低減できます。

- セキュリティへの影響を考慮する

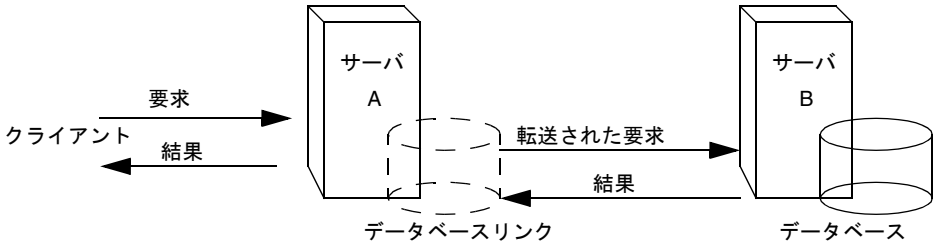
  アクセス制御はレフェラルの境界を越えると機能しません。要求を発信したサーバがエントリへのアクセスを許可している場合でも、スマートレフェラルがクライアントの要求を別のサーバに転送したときは、クライアントアプリケーションがアクセスを許可されないこともあります。

  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上でクライアント資格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に中継する手法の 1 つです。この手法はデータベースリンクを介して実行されます。80 ページの「データの分散」で説明したように、データベースリンク (database link) にデータは含まれていません。データベースリンクは、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に転送します。

連鎖 (chaining) では、サーバがそのサーバに格納されていないデータの要求を受け取ると、データベースリンクを使用してクライアントアプリケーションの代わりに別のサーバに接続し、結果をクライアントアプリケーションに返します。次の図に、この動作を示します。



各データベースリンクはデータを保持しているリモートサーバに関連付けられています。障害が発生したときにデータベースリンクが使用する、データの複製が入った代替リモートサーバを設定することもできます。データベースリンクの設定方法については、『iPlanet Directory Server 管理者ガイド』を参照してください。

データベースリンク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

  データベースリンクがクライアント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。データベースリンクは、エントリがディレクトリに再分散されるまで、レ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。接尾辞を使用してこの機能を実装することもできます。接尾辞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、レフェラルを返します。

- アクセス制御

データベースリンクは、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、リモートサーバに対するユーザ代行機能を無効にすることができます。データベースリンクの設定方法については、『iPlanet Directory Server 管理者ガイド』を参照してください。

レ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。

レ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インテリジェンスが置かれている場所です。連鎖システムでは、このインテリジェンスがサーバ内に実装されます。レフェラルを使用するシステムでは、インテリジェンス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、その分サーバ側の処理が複雑になります。連鎖対象のサーバは、リモートサーバ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。

レフェラルでは、クライアントがレ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、連鎖よりもレ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にフィードバックできます。

次に、レ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

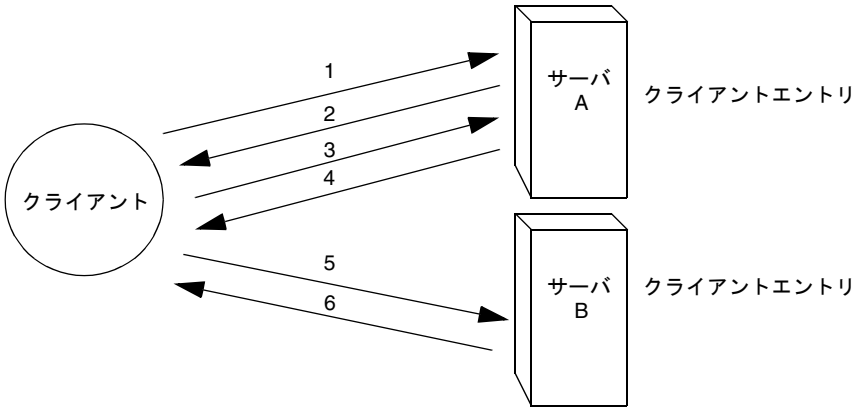
レフェラルをサポートしないクライアントアプリケーションもあります。連鎖を使用すると、クライアントアプリケーションは 1 つのサーバと通信するだけで、多数のサーバに格納されているデータにアクセスすることができます。企業のネットワークがプロキシを使用している場合は、レフェラルが機能しないことがあります。たとえば、クライアントアプリケーションがファイアウォール内の 1 つのサーバとだけ通信する権限を持っているとします。クライアントアプリケーションが別のサーバに転送された場合、クライアントはそのサーバに正常に接続できません。

また、レフェラルでは、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にクライアント資格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、レフェラルとは違う方法でアクセス制御を評価します。レフェラルでは、クライアントのエントリがすべての転送先サーバ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 A に送信する場合を考えてみます。次の図は、レフェラルを使用した場合の動作です。

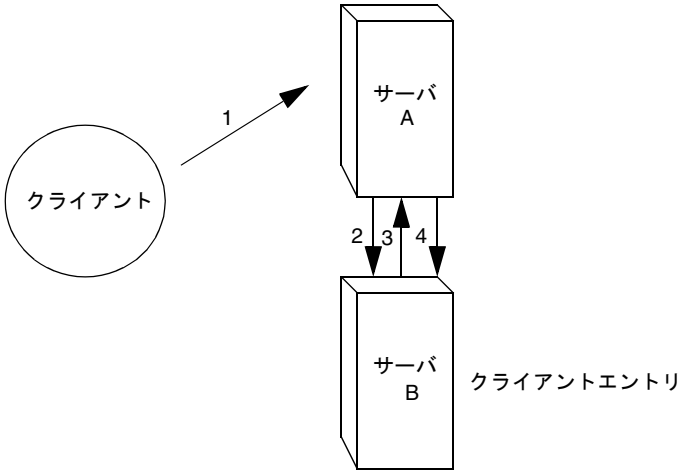


上記の図では、クライアントアプリケーションは次の手順を実行します。

1. クライアントアプリケーションは、まずサーバ A にバインドします。
2. サーバ A は、ユーザ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。レフェラルが機能するためには、サーバ A にクライアントエントリが存在していなければなりません。
3. クライアントアプリケーションがサーバ A に操作要求を送信します。
4. 要求された情報はサーバ A にないので、サーバ A はレ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 B に接続するように通知します。
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 B に送信します。サーバがバインドに成功するには、サーバ 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 B に送信できます。

この方法では、サーバ 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 B に必要です。

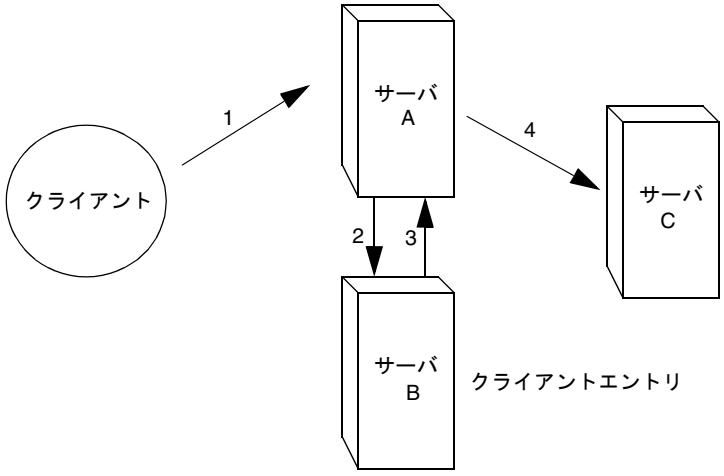
連鎖では、この問題は発生しません。連鎖システムでは、検索要求は次のように動作します。



上記の図では、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 A にバインドし、サーバ A はユーザ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 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 A にはクライアントの実際のエントリがあるサーバ B へのデータベースリンクがあります。サーバ A はバインド要求をサーバ B に送信します。
3. サーバ B はサーバ 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 A はデータベースリンクを使用してクライアントアプリケーションの要求を処理します。データベースリンクはサーバ 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上にある必要はありません。たとえば、システムを次のように設定できます。



上記の図では、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 A にバインドし、サーバ A はユーザ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 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 A にはクライアントの実際のエントリがあるサーバ B へのデータベースリンクがあります。サーバ A はバインド要求をサーバ B に送信します。
3. サーバ B はサーバ 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 A は別のデータベースリンクを使用してクライアントアプリケーションの要求を処理します。データベースリンクはサーバ 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、データベースリンクは、次のアクセス制御をサポートしません。

- ユーザエントリが別のサーバ上にある場合、そのユーザ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ない。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る
- 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る。これは、データベースリンクがリモートサーバに接続するときは、クライアントとして接続するためである。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、データベースリンクのドメインを使用してアクセス制御を評価する

インデックスを使用したデータベース性能の向上

データベースのサイズによっては、クライアントアプリケーションが実行する検索に多大な時間と資源が費やされることがあります。インデックスを使用すると、検索の性能を向上させることができます。

インデックスとは、ディレクトリのデータベースに格納されるファイルです。データベースごとに、個別のインデックスファイルがディレクトリ内に保持されています。各ファイルは、それがインデックス付けを行う属性に従って命名されます。特定の属性のインデックスファイルには複数のタイプのインデックスを含めることができるので、各属性について複数のタイプのインデックスを保持できます。たとえば、`cn.db3` というファイルには共通名属性のすべてのインデックスが入っています。

ディレクトリを使用するアプリケーションのタイプに応じて、さまざまなタイプのインデックスを使用します。アプリケーションの中には、特定の属性を頻繁に検索したり、異なる言語でディレクトリを検索したり、あるいは特定の形式のデータを必要とするものがあります。

この節では、次の事項について説明します。

- ディレクトリのインデックスタイプの概要
- インデックス付けのコスト評価

ディレクトリのインデックスタイプの概要

ディレクトリは、次のタイプのインデックスをサポートします。

- 実在インデックス

  実在インデックス (presence index) は、`uid` などの特定の属性を持つエントリをリストします。

- 等価インデックス

  等価インデックス (equality index) は、`cn=Babs Jensen` のような特定の属性値を持つエントリをリストします。

- 近似インデックス

  近似インデックス (approximate index) は、近似 (音による) 検索を可能にします。たとえば、`cn=Babs L. Jensen` という属性値を持つエントリがあるとします。`cn~=Babs Jensen`、`cn~=Babs`、および `cn~=Jensen` のいずれで検索しても、近似検索はこの cn=Babs L.Jensen という値を返します。

  近似インデックスは、ASCII 文字で書かれた英語名にだけ有効であることに注意してください。

- 部分文字列検索

部分文字列インデックス (substring index) は、エントリ内の部分文字列の検索を可能にします。たとえば、cn=*derson を検索すると、Bill Anderson、Norma Henderson、Steve Sanderson など、この文字列が含まれる共通名が検索されます。

- 国際化インデックス

  国際化インデックス (international index) は、国際化ディレクトリでの情報検索を高速化します。インデックスが付けられている属性にロケール (OID) を関連付けることで、マッチング規則が適用されるようにインデックスを設定できます。

- ブラウズインデックス

  ブラウズインデックス、あるいは仮想リスト表示 (VLV) インデックスは、Directory Server Console でのエントリの表示を高速化します。ディレクトリツリー内の任意の分岐にブラウズインデックス (browsing index) を作成することにより、表示性能を向上させることができます。

インデックス付けのコスト評価

インデックスはディレクトリデータベースでの検索性能を向上させますが、次のようなマイナス面もあります。

- インデックスが付けられていると、エントリの変更に時間がかかる

  保持するインデックスが増えると、ディレクトリがデータベースを更新するのにかかる時間もそれだけ長くなります。

- インデックスファイルがディスクスペースを占有する

  インデックスを付けた属性が増えると、作成するファイルの数も増えます。また、長い文字列を含む属性に近似インデックスや部分文字列インデックスを作成すると、ファイルはすぐに大きくなります。

- インデックスファイルはメモリを使用する

  ディレクトリは、実行効率を上げるため、できるかぎり多くのインデックスファイルをメモリ上に置きます。インデックスファイルは、データベースのキャッシュサイズに応じて使用可能なメモリプールを使用します。インデックスファイルの数が増えれば、データベースのキャッシュサイズも大きくする必要があります。

- インデックスファイルは作成に時間がかかる

  インデックスファイルは検索時間を短縮しますが、必要のないインデックスを保持すると、時間の浪費につながります。ディレクトリを使用するクライアントアプリケーションが必要とするファイルだけを保持するようにします。

第 6 章

レプリケーションの設計

ディレクトリの内容を複製すると、ディレクトリの可用性と性能が向上します。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 (topology) 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 (replication) 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。

またこの章では、複製の使用方法についても説明し、個々のディレクトリ環境に合った複製方法を設計するための指針も示します。この章は、次の節で構成されています。

- レプリケーションについて
- 一般的なレプリケーションの例
- レプリケーション戦略の定義
- 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、それ自身のデータベー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間でコピーできます。情報のマスターコピーを保持している Directory Server は、更新内容をすべてのレプリカに自動的にコピーします。

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

ディレクトリツリーを複数のサーバ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。クライアントは、読み取りや書き込み操作を実行するために、別の Directory Server に転送されます。書き込みフェイルオーバをサポートするには、多重マスター複製環境にする必要があります。

- 負荷均衡

  ディレクトリツリーを複数のサーバに複製することで、各マシン上のアクセス負荷を軽減し、サーバの応答時間を改善できます。

- 性能の向上と応答時間の短縮

  ディレクトリのエントリをユーザに近い場所に複製することで、ディレクトリの応答時間を大幅に改善できます。

- 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。

ディレクトリ情報の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。以下に、次の事項について説明します。

- 100 ページの「レプリケーションの概念」
- 105 ページの「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導入を計画するときは、常に次の基本事項を決定することから始めます。

- 複製する情報
- 情報のマスターコピーまたはサプライヤレプリカ (supplier replica) を保持するサーバ (複数も可)
- 情報の読み取り専用コピーまたはコンシューマレプリカ (consumer replica) を保持するサーバ (複数も可)
- コンシューマレプリカを保持するマシン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、その要求を転送する転送先サーバ

これらの事項は、Directory Server がこれらの概念をどのように処理するかを理解していないと効果的に決定できません。たとえば、レプリケーション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 で使用される概念を定義します。ここでは、全体的な決定事項について検討するための枠組みを示します。

レプリカ

レプリケーションに関与するデータベースのことを、レプリカと呼びます。レプリカにはいくつかの種類があります。

- マスターレプリカ : 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は処理できるが、更新要求はマスターレプリカに転送する
- 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、ハブレプリカはハブサプライヤとして動作する Directory Server に格納される

複数のデータベースを管理するように Directory Server を構成することができます。各データベースはレプリケーションにおける異なる役割を持つことができます。たとえば、マスターレプリカに `dc=engineering,dc=siroe,dc=com` 接尾辞を格納し、コンシューマレプリカに `dc=sales,dc=siroe,dc=com` 接尾辞を格納する Directory Server を構成できます。

サプライヤとコンシューマ

ほかのサーバに複製するマスターレプリカを管理するサーバは、サプライヤ (supplier) サーバまたはマスターサーバと呼ばれます。別のサーバによって更新されるコンシューマレプリカを管理するサーバは、コンシューマ (consumer) サーバと呼ばれます。

サーバのサプライヤまたはコンシューマとしての役割について説明します。ただし、サーバはサプライヤとコンシューマのどちらにもなれるため、厳密なものではありません。これは、以下のような場合に当てはまります。

- Directory Server 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
- Directory Server がハブサプライヤ (hub supplier) の役目を果たす場合。つまり、マスターサーバからの更新を受け取り、変更をコンシューマサーバにレプリケートする場合。詳細は、109 ページの「カスケード型レプリケーション」を参照
- マルチマスターレプリケーションで、一方の Directory Server が他方の Directory Server 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 上にマスターレプリカが保持される場合。詳細は、108 ページの「マルチマスターレプリケーション」を参照

iPlanet Directory Server 5.1 では、レプリケーションは常にサプライヤサーバから開始されます。コンシューマサーバから開始されることはありません。この処理は、サプライヤ主導レプリケーション (supplier-initiated replication) と呼ばれます。この処理により、1 つ以上のコンシューマサーバへデータをプッシュするようにサプライヤサーバを構成できます。

iPlanet Directory Server の旧バージョンでは、コンシューマサーバがサプライヤサーバからデータを引き出すコンシューマ主導レプリケーション (consumer-initiated replication) を開始することができました。iPlanet Directory Server 5.1 ではこれが変更され、コンシューマサーバがサプライヤサーバに更新の送信を促すようになりました。

すべての複製に対して、サプライヤサーバは次のような処理を実行する必要があります。

- ディレクトリクライアントからの読み取り、追加、および変更の要求に応答する
- 複製の状態情報と更新履歴ログを保持する
- コンシューマサーバへの複製を開始する

  サプライヤサーバは、管理しているサプライヤレプリカへの変更を常に記録します。そのため、すべての変更がコンシューマサーバにレプリケーションされます。

コンシューマサーバは次のような処理を実行する必要があります。

- 読み取り要求に応答する
- 追加要求と変更要求をレプリカのサプライヤサーバに転送する

  コンシューマサーバは、エントリの追加、削除、または変更の要求を受け取った場合、それらの要求を常にレプリカのサプライヤサーバに転送します。サプライヤサーバは要求を実行してから、変更を複製します。

カスケード型レプリケーションの場合、ハブサプライヤ (hub supplier) は次のような処理を実行する必要があります。

- 読み取り要求に応答する
- 追加要求と変更要求をレプリカのサプライヤサーバに転送する
- コンシューマサーバへの複製を開始する

カスケード型レプリケーションについては、109 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。

更新履歴ログ

すべてのサプライヤサーバは、更新履歴ログ (change log) を保持しています。更新履歴ログとは、サプライヤレプリカに対して行われた変更を記述しておく記録のことです。サプライヤサーバは、コンシューマサーバに格納されているレプリカに対して、またはマルチマスターのレプリケーションの場合はほかのマスターに対して、これらの変更を適用します。

エントリの変更、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。

以前のバージョンの Directory Server では、LDAP から更新履歴ログにアクセスできました。最新バージョンでは、サーバによる内部処理専用になりました。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のある場合は、レトロログのプラグインを使用して、下位互換性を保つ必要があります。詳細は、『iPlanet Directory Server 管理者ガイド』を参照してください。

レプリケーションの単位

iPlanet Directory Server 5.1 では、複製の最小単位はデータベースです。つまり、データベース全体を複製することはできますが、データベース内のサブツリーだけを複製することはできません。そのため、ディレクトリツリーを作成するときは、複製計画を考慮に入れる必要があります。ディレクトリツリーの設定方法については、第 5 章「ディレクトリトポロジの設計」を参照してください。

複製メカニズムでは、接尾辞とデータベースが 1 対 1 で対応している必要があります。つまり、カスタム分散論理を使用している 2 つ以上のデータベースにまたがって分散されている接尾辞 (またはネームスペース) は複製できません。

レプリケーションアグリーメント

Directory Server では、レプリケーションアグリーメントを使用して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 (replication agreement) は、1 つのサプライヤ (supplier) と 1 つのコンシューマ (consumer) との間で行われるレプリケーションを記述します。契約はサプライヤサーバ上に設定され、次のものを特定します。複製契約では、次のものを確認します。

- 複製するデータベース
- データが複製されるコンシューマサーバ
- レプリケーションを実行できる時間帯
- サプライヤサーバがコンシューマサーバにバインドするために使用する必要のある DN と資格 (レプリケーションマネージャエントリ、またはサプライヤバインド DN と呼ばれる。詳細は、104 ページの「レプリケーションの識別情報」を参照)

- 接続の安全性を確保するための手段 (SSL、クライアント認証)

レプリケーションの識別情報

2 つのサーバ間でレプリケーションが発生する場合は、サプライヤサーバ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サーバにバインドすると、コンシューマサーバはそのサプライヤサーバを認証します。この認証処理では、サプライヤサーバがコンシューマサーバにバインドするために使用するエントリが、コンシューマサーバに格納さ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、またはサプライヤバインド DN と呼ばれます。

レプリケーションマネージャエントリおよびその役割を遂行するために作成したエントリは、以下の条件を満たしていなくてはなりません。

- コンシューマレプリカ (またはハブレプリカ) を管理するすべてのサーバに、このエントリが少なくとも 1 つあること
- セキュリティ上の理由から、このエントリがレプリケーションされたデータの一部ではないこと

注 このエントリには、コンシューマサーバに定義されているアクセス制御規則をすべて無視する特別なユーザプロファイルがあります。

2 つのサーバ間でレプリケーションを構成する場合は、両方のサーバでレプリケーションマネージャエントリ (サプライヤバインド DN) を特定する必要があります。

- コンシューマサーバまたはハブサプライヤでは、コンシューマレプリカまたはハブレプリカを構成するときに、このエントリをレプリケーションの更新を実行する権利を持っているエントリとして指定する
- サプライヤサーバでは、レプリケーションアグリーメントを構成するときに、レプリケーションアグリーメントでこのエントリの DN を指定しておく

注 Directory Server Console では、このレプリケーションマネージャエントリがサプライヤバインド DN と呼ばれるため、このエントリがサプライヤサーバには存在しないという誤解を産むことがあります。サプライヤバインド DN と呼ばれるのは、これが、サプライヤサーバ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サーバにバインドしたときに、コンシューマサーバがサプライヤサーバを認証できるように、コンシューマサーバに存在していなければならないエントリであるからです。

データの整合性

整合性とは、複製されたデータベースの内容が、任意の時点でどの程度マッチしているかを示します。2 つのサーバ間で複製を設定する場合は、設定の中で更新をスケジュールします。iPlanet Directory Server 5.1 では、必ずサプライヤサーバがいつコンシューマサーバ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。

Directory Server には、常に複製の同期を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。常に複製の同期を維持する利点は、データの整合性がより確実に保証されるという点です。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、次の場合に最適です。

- サーバ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、必要に応じて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、次の場合に最適です。

- 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 (ダイアルアップ接続を使用して複製の同期をとっている場合など)
- ディレクトリがサービスす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、各マスター上の複製は緩やかな 整合性を保っている状態といえます。これは、常に複製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の例

レプリケーションの更新情報をサーバ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間の相互動作の方法を決める必要があります。次の 3 つの基本的な例について説明します。

- 単一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

次に、上記の複製方法について説明し、環境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。これらの基本的な例を組み合わせて、要件にもっとも合ったレプリケーショントポロジを構築することもできます。

単一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構成では、1 つのマスターサーバが 1 つ以上のコンシューマサーバにサプライヤレプリカを直接コピーします。この構成では、ディレクトリの変更はすべてサプライヤサーバ上のサプライヤレプリカ (supplier replica) 上で行われ、コンシューマサーバにはデータの読み取り専用コピーが格納されています。

サプライヤサーバは、マスターレプリカへのすべての変更を記録する更新履歴ログを維持します。また、サプライヤサーバにはレプリケーションアグリーメントも格納されます。

サプライヤサーバ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サーバにバインドしたときに、コンシューマサーバがサプライヤサーバを認証できるように、コンシューマサーバにはサプライヤバインド DN に対応しているエントリが格納されます。

サプライヤサーバはすべての変更をコンシューマレプリカに伝達する必要があります。107 ページの図 6-1 に、この簡単な構成を示します。

図 6-1　単一マスター複製

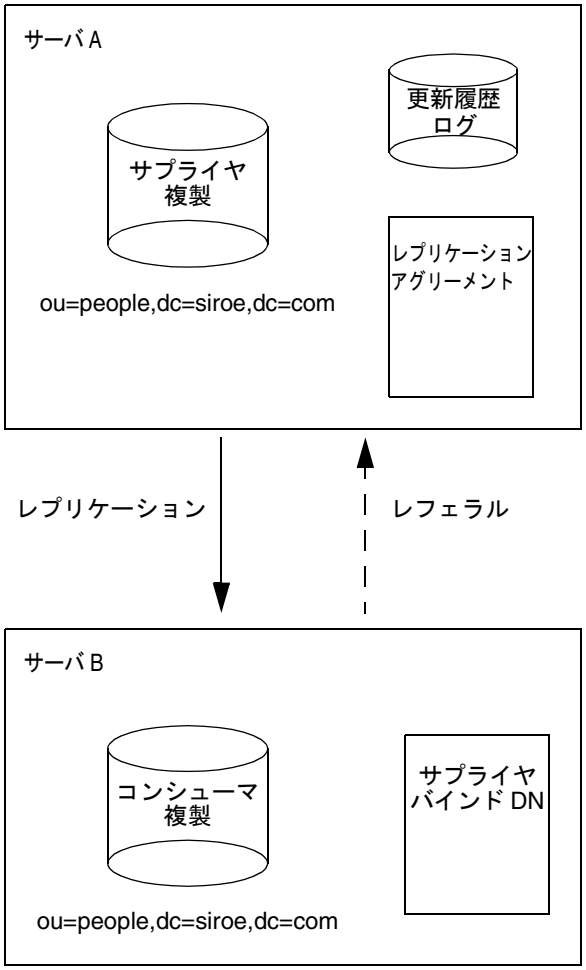


図 6-1 にはコンシューマサーバが 1 つしかありませんが、サプライヤサーバは複数のコンシューマサーバにレプリケーションできます。1 つのサプライヤサーバが管理できるコンシューマサーバの総数は、ネットワークの速度と 1 日当たりのエントリの変更総数によって異なりますが、1 つのサプライヤサーバで複数のコンシューマサーバを管理できると想定して特に問題ありません。

マルチマスターレプリケーション

多重マスター構成には、次の利点があります。

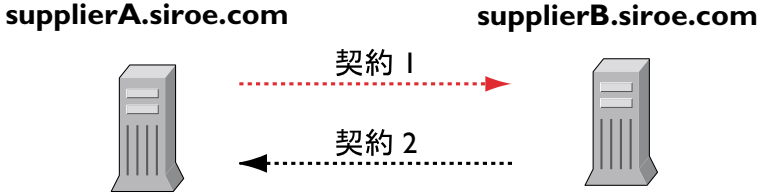
- 1 つのサプライヤにアクセスできなくなった場合でも、自動的に書き込み処理のフェイルオーバが実行される
- 地域分散型環境のローカルサプライヤで更新処理を実行できる

マルチマスターレプリケーション (multi-master replication) 環境では、同じ情報のマスターコピーが 2 つのサーバ上に存在します。これは、異なる場所でデータを同時に更新できるということを意味します。一方のサーバ上で行われた変更は他方のサーバに複製されます。つまり、各サーバはサプライヤとコンシューマの両方の役割を果たします。

同じデータが両方のサーバ上で変更された場合は、どちらの変更を格納するか決めるために、競合を解決するための措置がとられます。Directory Server は、最新の変更を有効な変更とみなします。

独立した 2 つのサーバ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 つの複製契約においては、1 つのサプライヤサーバと 1 つのコンシューマサーバだけしか存在できません。そのため、同じデータの管理責任を共有する 2 つのサプライヤサーバ間にマルチマスター環境を構築するには、複数のレプリケーションアグリーメントを作成する必要があります。次の図に、この設定を示します。

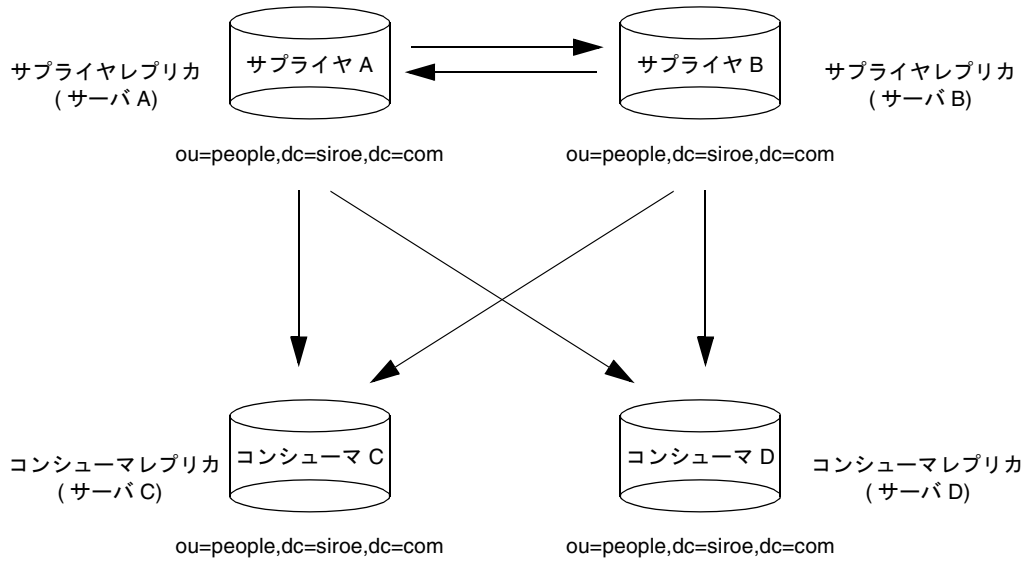
図 6-2　多重マスター複製の構成 (2 つのマスター)



この図では、サプライヤ A とサプライヤ B がそれぞれ同じデータのサプライヤレプリカを保持します。

保有できるマスターまたはサプライヤの数は、どの複製環境でも 2 つだけです。ただし、コンシューマレプリカを保持するコンシューマサーバの数は制限されていません。109 ページの図 6-3 に、2 つのマスターサーバと 2 つのコンシューマサーバが存在する環境でのレプリケーショントラフィックを示します。この図は、コンシューマが両方のマスターによって更新されることを示しています。マスターは、変更の競合を調整する措置をとります。

図 6-3　多重マスター環境での複製トラフィック



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は、以下のような場合に非常に役立ちます。

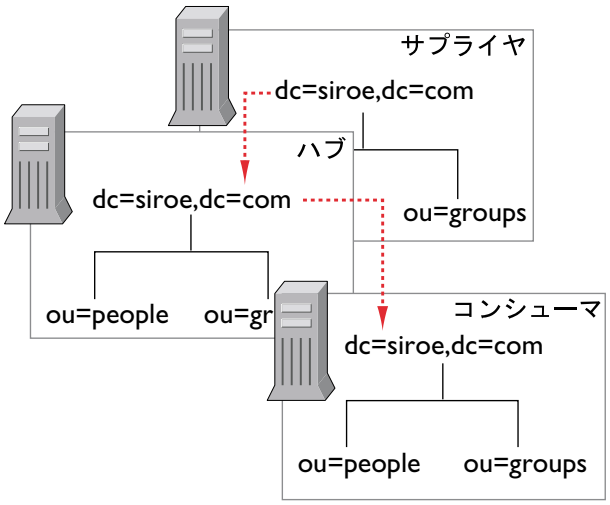
- 非常に大きなトラフィック負荷を均等にする必要がある場合。たとえば、サプライヤサーバはすべての更新トラフィックを処理する必要があるため、コンシューマサーバへのすべてのレプリケーショントラフィックをサポートするには、非常に大きな負荷がかかる。多数のコンシューマに対するレプリケーションの更新を処理できるハブサーバにレプリケーショントラフィックの負荷を移すことができる
- 地域分散型環境でローカルハブサプライヤを使用して、接続コストを削減する場合
- ディレクトリサービスの性能を向上させる場合。読み取り操作を行うすべてのクライアントアプリケーションをコンシューマへ、更新操作を行うすべてのクライアントアプリケーションをサプライヤへ導くことができるなら、ハブサーバのすべてのインデックス (システムインデックスを除く) を削除できる。これによって、サプライヤとハブサーバとの間のレプリケーション速度が大幅に向上する

カスケード型レプリケーション (cascading replication) では、ハブサプライヤがサプライヤサーバから更新情報を受け取り、コンシューマサーバに更新内容を適用します。ハブサプライヤ (hub supplier) はハイブリッドです。つまり、通常のコンシューマサーバ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、通常のサプライヤサーバと同様に更新履歴ログも保持しています。

ハブサプライヤは、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転送します。同様に、ディレクトリクライアントから追加または変更の要求を受け取ったときは、そのクライアントをマスターサーバに転送します。

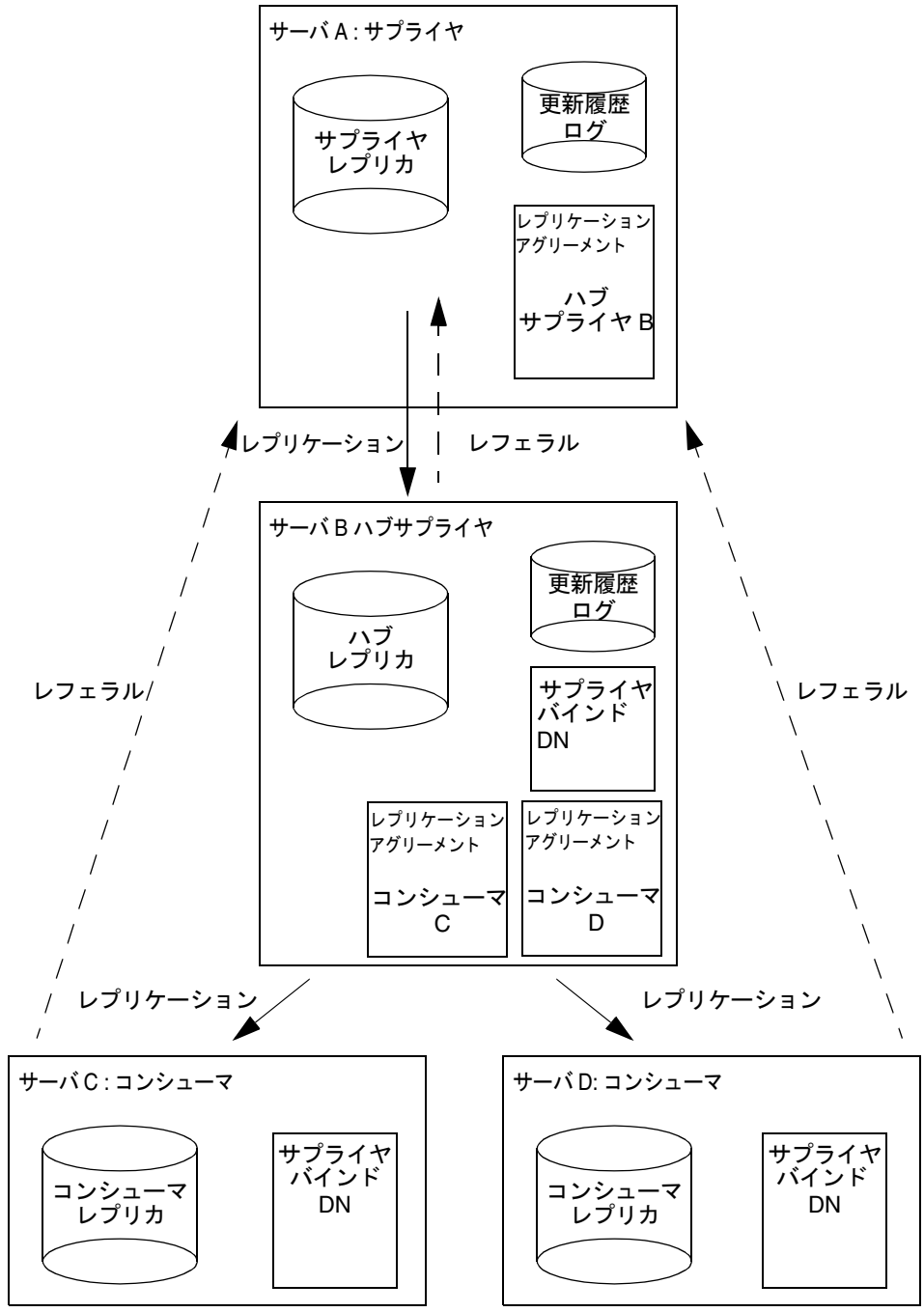
このカスケード型複製の例を図 6-4 に示します。

図 6-4　カスケード型レプリケーションの例



同じ例を別の視点から図にすると、図 6-5 のようになります。この図では、サーバの構成を示します (レプリケーションアグリーメント、更新履歴ログ、レフェラル)。

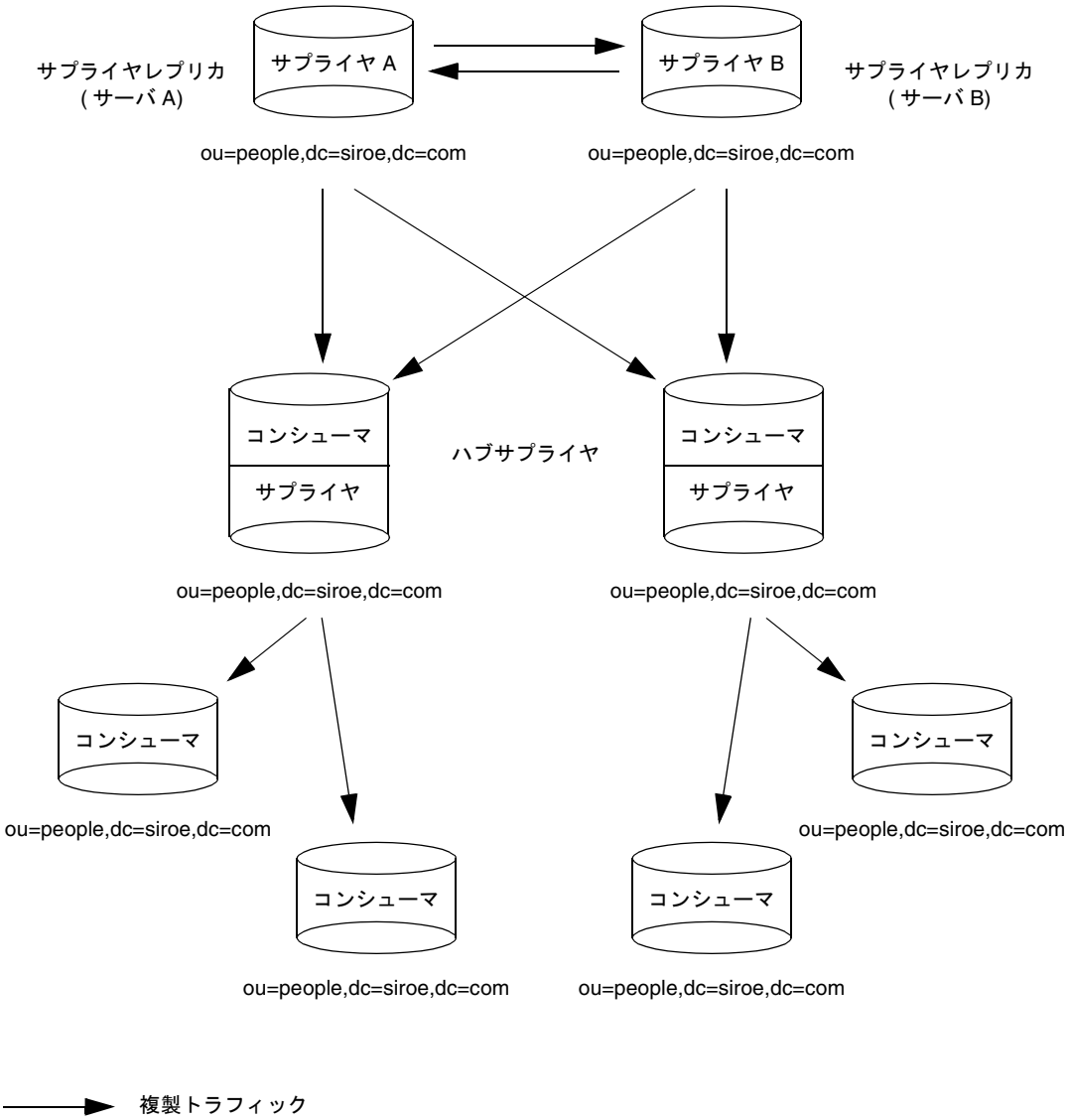
図 6-5　カスケード型レプリケーションでのサーバ構成



混合環境

上記で説明した例を任意に組み合わせて、要件にもっとも合った環境を構築できます。たとえば、マルチマスター構成とカスケード型構成を組み合わせて、113 ページの図 6-6 に示すような構成も可能です。

図 6-6　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



レプリケーション戦略の定義

使用する複製手法は、提供するサービスによって決まります。

- 高可用性を最優先する場合は、1 つのサイトに複数の Directory Server を配備したデータセンターを構築する必要がある。読み取りフェイルオーバを提供するには単一マスター複製を、書き込みフェイルオーバを提供するには多重マスター複製を使用できる。高可用性を実現するためのレプリケーションの設定方法については、116 ページの「高可用性を実現するためのレプリケーションの使用」で説明している
- ローカルでのデータの利用性を最優先する場合は、複製を使用して、世界中のオフィスに配置されている Directory Server にデータを物理的に分散する必要がある。本社などの単一の場所にすべての情報のマスターコピーを保管するか、あるいはそれぞれのローカルサイトが自分のサイトに関連する DIT 部分を管理するようにするか選択できる。レプリケーションの設定タイプについては、117 ページの「ローカルでのデータの利用性を高めるためのレプリケーションの使用」で説明している
- どの場合でも、Directory Server がサービスする要求の負荷を均等にして、ネットワークへの過剰負荷を防ぐ必要がある。Directory Server とネットワークへの負荷を均等にする戦略については、117 ページの「負荷均衡のためのレプリケーションの使用」で説明している

複製手法を決定するには、ネットワーク、ユーザ、アプリケーション、および提供するディレクトリサービスがユーザやアプリケーションによってどのように使われるかを調査することから始めます。この調査のガイドラインについては、「レプリケーション調査」を参照してください。

複製手法を決定したら、ディレクトリの導入を開始できます。ディレクトリサービスを段階的に導入していくことをお勧めします。ディレクトリを実際の環境に導入する段階に入ると、ディレクトリを全体的に導入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、複製方法の決定に影響する要因について詳しく説明します。

- 115 ページの「レプリケーション調査」
- 115 ページの「レプリケーション資源の要件」
- 116 ページの「高可用性を実現するためのレプリケーションの使用」
- 117 ページの「ローカルでのデータの利用性を高めるためのレプリケーションの使用」
- 117 ページの「負荷均衡のためのレプリケーションの使用」
- 121 ページの「小規模サイト向けのレプリケーション方法の例」

- 121 ページの「大規模サイト向けのレプリケーション方法の例」

レプリケーション調査

複製手法を決定するときは、調査を通じて次のような情報を収集する必要があります。

- 異なる建物間やリモートサイト間を接続する LAN および WAN の品質と使用可能な帯域幅の大きさ
- ユーザの物理的な位置、各サイトのユーザ数、ユーザのアクティビティ

  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知る必要があります。ディレクトリを使用するほかの認証アプリケーションや META Directory アプリケーションなど、典型的な製品です。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数とサイズ

レプリケーション資源の要件

複製を使用する場合は、さらに資源が必要になります。複製手法を決定するときは、次の資源要件を検討してください。

- ディスク使用量

  サプライヤサーバでは、各更新操作後に更新履歴ログが書き込まれます。サプライヤサーバが多数の updabhubcon2supplier サーバを受け取ると、サプライヤサーバに複数の複製データベースがある場合は、更新履歴ログがより頻繁に使用され、ディスクの使用量がさらに増大します。
- サーバスレッド

  複製契約ごとに 1 つのサーバスレッドが消費されます。そのため、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の性能に影響があります。
- ファイルディスクリプタ

サーバ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) と各複製契約 (契約ごとに 1 ファイルディスクリプタが必要) によって減少します。

高可用性を実現するためのレプリケーションの使用

複製を使用して、単一のサーバ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します。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に複製しておきます。

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 回は複製しておく必要があると主張する人もいます。耐障害性を目的とした複製の使用頻度はユーザの決定事項ですが、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の代替として複製を使用してはいけません。ディレクトリデータのバックアップについては、『iPlanet Directory Server 管理者ガイド』を参照してください。

すべてのディレクトリクライアントに書き込みフェイルオーバを保証する必要がある場合は、マルチマスターレプリケーションの手法を使用する必要があります。読み取りフェイルオーバで充分な可用性が達成できる場合は、単一マスター複製を使用できます。

LDAP クライアントアプリケーションは通常、1 つの LDAP サーバだけを検索するように設定できます。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 サーバを循環するように作成していないかぎり、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。

代わりに、iPlanet Directory Access Router 製品を使用することもできます。iPlanet Directory Access Router については、http://www.iplanet.com を参照してください。

ローカルでのデータの利用性を高めるためのレプリケーションの使用

複製を使用して、ローカルでもデータを利用できるようにする必要があるかどうかはネットワークの品質とそのサイトでなにを行うかによって決まります。また、ディレクトリに格納するデータの特性と、データが一時的に使用できなくなった場合の会社への影響について慎重に考慮する必要があります。データの重要性が高ければ、それだけ品質の低いネットワーク接続によって引き起こされる業務停止は、許容できないものになります。

次の理由により、ローカルでもデータを利用するために複製を使用する必要があります。

- データのローカルマスターコピーが必要である

  これは、特定の国の従業員にだけ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な戦略です。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、経営方針として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重要です。

- 信頼性が低いネットワーク接続、または断続的に利用可能なネットワーク接続を使用している

  国際ネットワークでよく見られるように、信頼性の低い WAN 使用している場合はネットワーク接続を断続的に行います。

- ネットワークに過度な負担が定期的にかかり、ディレクトリの性能が著しく低下する

  たとえば、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。

負荷均衡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 にかかる負荷を均等にすることができます。

- ユーザの検索アクティビティを複数のサーバに分散する
- 書き込みはサプライヤサーバだけに限定し、その他のサーバは読み取り専用に設定する
- メールサーバのように、特定のタスクに専用サーバを割り当てる

ディレクトリデータを複製する重要な理由の 1 つとして、ネットワーク負荷の均衡が挙げられます。可能であれば、比較的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に、データを移動します。もっとも重要な点は、サーバとディレクトリユーザとの間のネットワーク接続の通信速度と信頼性です。

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が検索を 1 日当たり約 10 回実行すると、ネットワークの負荷はユーザ 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に複製する必要がある場合もあります。

データをローカルに利用できるという利点が、複製を使用したことによるネットワーク負荷の増加という不利益を上回るかどうか慎重に検討してください。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。

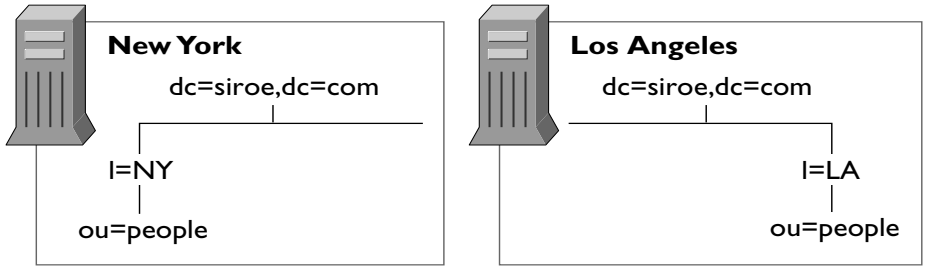
たとえば、ディレクトリツリーに平均して 1,000,000 件を超えるエントリがあり、毎日 10 % 前後が変更される場合を考えてみます。ディレクトリエントリのサイズが平均して 1K バイトとすると、ネットワークの負荷が 1 日当たり 100M バイト増えることになります。しかし、リモートサイトの従業員がたとえば 100 人と少なく、1 日当たり平均して 10 回のディレクトリ検索を実行している場合は、ディレクトリアクセスによるネットワークの負荷は 1 日当たり 1M バイトにすぎません。

複製による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、ネットワークの負荷均衡を目的とする複製は好ましく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複製を使用します。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、105 ページの「データの整合性」を参照してください。

ネットワークの負荷均衡の例

2 つの都市にオフィスを持つ企業を考えてみます。各オフィスには、次の方法で管理する特定のサブツリーがあります。

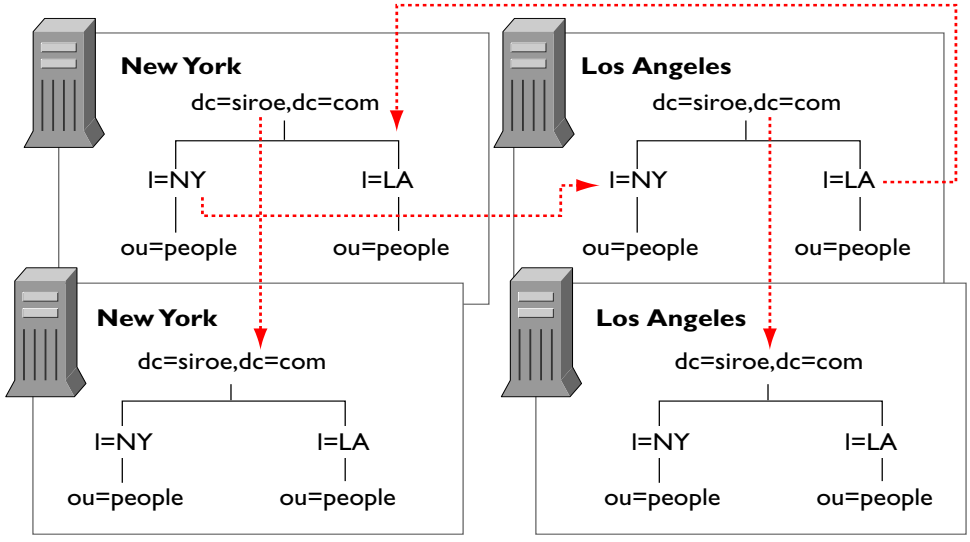


各オフィスには高速のネットワークがありますが、2 都市間の通信にはダイヤルアップ接続を使用しています。このような場合は、以下のようにネットワークの負荷を均等にします。

- オフィスごとに、ローカルで管理するデータのマスターサーバになるサーバを 1 つ選択する

  ローカルで管理するデータを、選択したサーバからリモートオフィスのマスターサーバに複製します。

- ディレクトリデータ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、各マスターサーバ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 にレプリケーションする。ローカルで管理する接尾辞にはマルチマスターレプリケーションを使用でき、リモートサーバからデータのマスターコピーを受け取る接尾辞にはカスケード型レプリケーションを使用できる



性能向上のための負荷均衡の例

ディレクトリで 1,500,000 のエントリを格納して 1,000,000 のユーザをサポートする必要があり、各ユーザが 1 日当たり 10 件のディレクトリ検索を実行する場合を考えてみます。また、1 日当たり 25,000,000 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するメッセージングサーバを使用しているとします。この場合、メール処理に 1 日当たり 125,000,000 件のディレクトリ検索が実行されると予測できます。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 135,000,000 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、1,000,000 人のディレクトリユーザが 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、1 日 12 時間で 135,000,000 件のディレクトリ検索をサポートする必要があります。これは毎秒 3,125 (135,000,000 / (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。

1,000,000 人のユーザ	1 ユーザにつき 10 件の検索 =	10.000.000 件の読み取り / 日
25,000,000 通のメッセージ	1 メッセージにつき 5 件の検索 =	125,000,000 件の読み取り / 日
	合計読み取り / 日 =	135,000,000
12 時間は 43,200 秒	合計読み取り / 秒 =	3,125

ここで、毎秒 500 件の読み取りをサポートできる CPU と RAM の組み合わせを Directory Server で使用しているとします。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。ただし、1,000,000 人のディレクトリユーザを有する企業の場合は、ローカルでデータを利用することも考慮するとさらに Directory Server を追加する必要があります。

これらの計算から、次のような方法で複製します。

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

  この構成は、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。

- 上記のサプライヤサーバを使用して 1 つ以上のハブサプライヤに複製する

  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サーバで処理されるので、マスターサーバは書き込み要求の処理に専念できます。ハブサプライヤの定義については、109 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。

- ハブサプライヤを使用して、会社全体のローカルサイトに複製する

  ローカルサイトにレプリケーションすることにより、サーバや WAN の作業負荷を均等にし、ディレクトリデータを高可用性を得るようにすることが可能です。国内の 4 つのサイトにレプリケーションする場合を考えてみます。その場合、各ハブサプライヤに 4 つのコンシューマが存在することになります。

- 各サイトで、少なくとも 1 回はレプリケーションして高可用性を得るようにする。また、最低でも読み取り操作を実行できることを確認する

  DNS ソートを使用して、ローカルユーザ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方法の例

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置されています。ネットワークは非常に安定しており、サーバのハードウェアと 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 回はレプリケーションしておきます。また、Directory Server の 1 つが使用できなくなったときに LDAP 接続の性能を上げるために、DNS ラウンドロビンを設定します。代替方法として、iPlanet Directory Access Router などの LDAP プロキシを使用することもできます。iPlanet Directory Access Router については、http://www.iplanet.com 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方法の例

会社が 2 つの建物に分かれている場合を考えてみます。各建物には、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。ネットワークは非常に安定しており、サーバのハードウェアと 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 つのサーバの処理能力で、各建物内のサーバ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。

複製方法は次のようになります。

- 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 つのサーバを選択する

このサーバは、ディレクトリ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。

- ディレクトリ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 回は複製する

  書き込みフェイルオーバを保証する必要がある場合は、多重マスター複製構成を使用します。

- もう一方の建物 (建物 B) 内に 2 つの複製を作成する
- サプライヤサーバとコンシューマサーバと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

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 iPlanet 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、最適な複製の設計に役立つ、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 ACI をエントリの属性として格納しています。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ションされます。Directory Server は ACI をローカルに評価するので、このことは重要です。

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 章の 127 ページの「安全なディレクトリの設計」を参照してください。

レプリケーションと Directory Server のプラグイン

レプリケーションは、iPlanet Directory Server に付属するほとんどのプラグインと一緒に使用できます。多重マスター複製で次のプラグインを使用する場合には、いくつかの例外と制限があります。

- 属性一意性検査プラグイン
- 参照整合性検査プラグイン

マルチマスターレプリケーションと属性一意性検査プラグインは一緒に使用できません。これは、このプラグインが、マルチマスターセット内の両方のサーバではなく、同じサーバ上の属性値しか検証できないためです。

参照整合性検査プラグインが多重マスターセット内の 1 つのマスター上でだけ有効になっている場合は、このプラグインと多重マスター複製を一緒に使用できます。これにより、参照整合性の更新は必ず 1 つのマスターサーバ上だけで行われ、それがほかのサーバに伝達されることが保証されます。

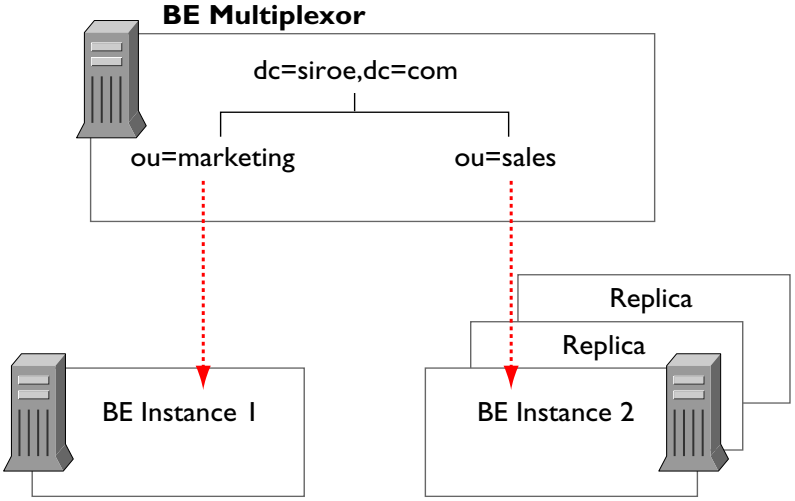
注	デフォルトでは、これらのプラグインは無効になっています。有効にするには、Directory Server Console またはコマンド行ユーティリティを使用する必要があります。

レプリケーションとデータベースリンク

連鎖 (chaining) を使用してエントリを分散する場合は、データベースリンク (database link) を含むサーバが、実際のデータがあるリモートサーバを指示します。このような環境では、データベースリンク自体を複製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データベースは複製できます。

レプリケーションをデータベースリンクのバックアップには使用しないでください。データベースリンク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 (entry distribution) については、79 ページの「ディレクトリトポロジの設計」を参照してください。

図 6-7　連鎖データベースのレプリケーション



スキーマのレプリケーション

レプリケーション環境で iPlanet 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 Directory Server 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多くの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環境の場合でも、1 つのマスターサーバでスキーマの変更を行うことをお勧めします。

スキーマの複製は自動的に行われます。サプライヤとコンシューマ間で複製が設定されている場合は、デフォルトでスキーマが複製されます。

iPlanet 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どのレプリケーションでも同じで、次のように説明できます。

1. データをコンシューマサーバに送る前に、サプライヤサーバは自身のスキーマがコンシューマサーバ上で保持されているスキーマ (schema) 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、複製が実行されます。
3. サプライヤサーバ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複製を実行する前に自身のスキーマをコンシューマに複製します。

注 サプライヤサーバ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古い場合は、コンシューマ上のスキーマが新しいデータをサポートできないので、複製中に多数のエラーが発生します。

マルチマスターセットの 2 つのマスターサーバでスキーマの変更を行う場合、コンシューマには 2 つのマスターからレプリケーションされた、それぞれが異なるスキーマを持つデータが格納されます。最後に更新されたマスターが優先され、そのスキーマがコンシューマに伝達されます。このため、コンシューマ上のスキーマは、常に一方のマスターのスキーマとは異なることになります。これを回避するために、常に 1 つのマスターだけでスキーマの変更を行います。

注 コンシューマサーバ上のスキーマは絶対に更新しないでください。サプライヤサーバは起こりうる不整合を解決できないので、レプリケーションは失敗します。

スキーマは複製トポロジのマスターサプライヤサーバ上で保持する必要があります。標準の 99user.ldif ファイルを使用している場合は、これらの変更がすべてのコンシューマに複製されます。カスタムのスキーマファイルを使用している場合は、マスターサプライヤ上で変更を行ったら、必ずこれらのファイルをすべてのサーバにコピーしてください。ファイルをコピーしたら、サーバを再起動する必要があります。詳細は、51 ページの「カスタムスキーマファイルの作成」を参照してください。

カスタムのスキーマファイルへの変更は、スキーマが LDAP または Directory Server Console を使用して更新されている場合にだけ複製されます。これらのカスタムスキーマファイルは、すべてのサーバ上で情報を同じスキーマファイルに保持するために、各サーバにコピーする必要があります。詳細は、51 ページの「カスタムスキーマファイルの作成」を参照してください。

スキーマ設計については、第 3 章「スキーマの設計」を参照してください。

第 7 章

安全なディレクトリの設計

Directory Server のデータを保護する方法は、これまで説明してきたすべての設計領域に影響します。セキュリティの設計は、ディレクトリに格納されているデータを保護し、ユーザとアプリケーションのセキュリティおよび機密性の要件を満たすものでなければなりません。

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について
- セキュリティ要件の分析
- セキュリティ手法の概要
- 適切な認証方式の選択
- アカウントの無効化による認証の防止
- パスワードポリシーの設計
- アクセス制御の設計
- SSL による接続のセキュリティ保護
- その他のセキュリティ関連資料

セキュリティに対する脅威について

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 (DoS)

次に、ディレクトリのセキュリティポリシーを設計するときに役立つように、一般的なセキュリティの脅威について説明します。

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、実際には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。

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資格を利用してデータにアクセスする場合があります。保護されていないパスワードをディレクトリで使用している場合は、この種の不正アクセスが行われることがよくあります。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も考えられます。

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。

ここに挙げた例は、権限のないクライアントからディレクトリデータにアクセスする例の一部にすぎません。

iPlanet 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。これらの問題に関しては、133 ページの「適切な認証方式の選択」、137 ページの「パスワードポリシーの設計」、および 142 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。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。

たとえば、ディレクトリが改ざんを検出できない場合、侵入者はクライアントからサーバへの要求を変更 (または遮断) したり、サーバからクライアントへの応答を変更したりすることが可能になります。SSL や類似の技術では、接続のどちらかで情報に署名することで、この問題を解決しています。iPlanet Directory Server での SSL の使い方については、151 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 (DoS)

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資源を消費して、ほかのユーザが資源を使用するのを妨害します。

iPlanet Directory Server では、特定のバインド DN に割り当てる資源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。ユーザのバインド DN に基づく資源制限の設定方法については、『iPlanet Directory Server 管理者ガイド』の「ユーザアカウントの管理」を参照してください。

セキュリティ要件の分析

自社のセキュリティ要件を特定するには、環境とユーザを分析する必要があります。第 3 章の「ディレクトリデータの設計」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が読み取りまたは書き込みができるかの基本的な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に解放されている e コマース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とアプリケーションに与える
- 社員と業務に関する機密データを通常のアクセスから保護する

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 e コマースアプリケーションをサポートしたりする場合は、上記の点に加えて、次の事項を考慮する必要があります。

- 顧客に機密性を保証する
- 情報の完全性を保証する

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。

- 130 ページの「アクセス権限の決定」
- 130 ページの「データの機密性と完全性の保証」
- 131 ページの「定期的な監査の実行」
- 131 ページの「セキュリティ要件の分析例」

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、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 とおりの方法でアクセス権限 (access rights) を与えます。

- 機密データを保護しながら、すべてのカテゴリのユーザに最大限のアクセスを与える

  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が必要とする情報、また場合によっては外部のユーザ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。

ユーザの識別方法については、133 ページの「適切な認証方式の選択」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、142 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 e コマース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。

- 転送データを暗号化する
- 証明書を使用して転送データに署名を付ける

iPlanet Directory Server で使用可能な暗号化方法については、140 ページの「パスワード保存スキーマ」を参照してください。データの署名については、151 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。

定期的な監査の実行

さらにセキュリティを高めるために、セキュリティポリシー全体の効率を検証する定期的な監査を実行する必要があります。定期的な監査では、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します。

SNMP については、『iPlanet Directory Server 管理者ガイド』を参照してください。

セキュリティ要件の分析例

ここでは、架空の ISP である siroe.com 社が自社のセキュリティ要件をどのように解析したかを例に示しながら説明します。

siroe.com 社の業務は、Web ホスティングとインターネットへのアクセスを提供することです。siroe.com 社では業務の一部として、クライアント企業のディレクトリをホストしています。また、多数の個人加入者にインターネットへのアクセスを提供しています。

このため、siroe.com 社はディレクトリ内に次の 3 つの主要な情報カテゴリを持ちます。

- siroe.com の内部情報
- 法人顧客に属する情報
- 個人加入者に関係する情報

siroe.com 社は、次のようなアクセス制御を必要とします。

- Company22 と Company33 のディレクトリ管理者に自社のディレクトリ情報へのアクセスを許可する
- Company22 と Company33 自身のアクセス制御ポリシーをそれらのディレクトリの情報について実装する
- siroe.com 社のサーバを使用して自宅からインターネットにアクセスするすべての個人クライアントについて、標準のアクセス制御ポリシーを実装する
- siroe.com 社のディレクトリへの外部からのアクセスをすべて拒否する
- siroe.com 社の加入者用ディレクトリに対する読み取り権限をすべてのユーザに与える

セキュリティ手法の概要

iPlanet Directory Server では、個々の要件に合ったセキュリティポリシーを設計するためのさまざまな手法が用意されています。セキュリティポリシーは、権限のないユーザ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。

iPlanet 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 にパスワードを与えます。

- 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に与えるアクセス権限を調整し、必要な資格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、すべての認証試行を自動的に拒否します。

- SSL による署名

  情報の完全性を保持します。情報が署名されている場合、受信者は情報が転送中に変更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、複製やデータの分散など、ほかのディレクトリの機能使用できます。

適切な認証方式の選択

セキュリティポリシーに関して決定が必要な項目に、ユーザのディレクトリへ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、またはディレクトリを使用するすべてのユーザ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。

iPlanet Directory Server では、次のような認証 (authentication) 方法を使用できます。

- 匿名アクセス
- 簡易パスワード
- 証明書に基づく認証
- TLS を介した簡易パスワード
- プロキシ認証

ディレクトリでは、相手がユーザか LDAP 対応アプリケーションかにかかわらず、すべてのユーザに対して同じ認証メカニズムを使用します。

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、137 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 (anonymous access) 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、特定のユーザまたはユーザ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 (書き込み、追加、削除、本人による書き込みは許可しません)。また、アクセスは、ユーザ名、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。

ユーザパスワード属性が含まれていないエントリでユーザ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。

- ユーザがパスワードを与えない場合は、匿名アクセスを与える

- パスワードに対してなんらかの文字列を与えた場合、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 -D "cn=joe" -w secretpwd -b "siroe.com" cn=joe
```

この場合、ディレクトリ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マッチ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ディレクトリ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、サーバへの認証を行います。

たとえば、識別名と資格のセットを送信するバインド操作によって、クライアントはディレクトリへの認証を行います。サーバはクライアント 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マッチするかどうかを確認します。マッチした場合、サーバはクライアントを認証します。マッチしない場合、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 (bind DN) はユーザのエントリに対応しています。ただし、ユーザのエントリとしてではなく組織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。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 をユーザに表示しません。クライアントがバインド DN をユーザに表示しない場合、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. ユーザはユーザ ID などの一意の識別子を入力する (たとえば、`fchen`)
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=fchen,ou=people,dc=siroe,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ネットワークに送信されることです。悪意を持ったユーザが盗聴している場合、認証されたユーザになりすます可能性があり、ディレクトリ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。

証明書に基づく認証

ディレクトリ認証のもう 1 つの方式として、ディレクトリへのバインドにセキュリティ証明書を使用する方法があります。ユーザがディレクトリに初めてアクセスすると、ディレクトリはユーザにパスワードを要求します。ただし、このパスワードは、ディレクトリに格納されているパスワードとマッチするものではなく、そのユーザの証明書 (certificate) データベースを開くためのパスワードです。

ユーザが入力したパスワードが正しい場合は、ディレクトリクライアントアプリケーションは証明書データベースから認証情報を取得します。次に、クライアントアプリケーションとディレクトリは、この情報を使用し、ユーザの証明書をディレクトリの DN に割り当てることでユーザを識別します。この識別過程で検出されたディレクトリ DN に基づいて、ディレクトリへのアクセスが許可または拒否されます。

証明書および SSL については、『Managing Servers with iPlanet Console』を参照してください。

TLS を介した簡易パスワード

SSL または「Start TLS」操作を使用して Directory Server とクライアントアプリケーションの間に安全な接続が確立された場合、サーバはパスワードを要求して特別なレベルの認証を要求することができます。この場合、パスワードがクリアテキストでネットワークに送信されることはありません。

SSL については、151 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。「Start TLS」操作については、『iPlanet Directory Server 管理者ガイド』を参照してください。

プロキシ認証

プロキシ認証 (proxy authorization) 方式は特殊な形式の認証で、ユーザは自分の ID を使用してディレクトリにバインドしますが、プロキシ認証によって別のユーザの権限が与えられます。

たとえば、プロキシ認証を使用すると、ディレクトリ管理者は一般ユーザとしてディレクトリ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資格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、一般ユーザの権限が与えられます。このような仮のユーザはプロキシユーザと呼ばれ、そのユーザの 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ディレクトリを構成するには、次の手順を実行します。

- 管理者には、ほかのユーザとしてのプロキシ権限を与える
- 一般ユーザには、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を与える

注 ディレクトリマネージャ以外のディレクトリのすべてのユーザ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 (ディレクトリマネージャ 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。

プロキシのメカニズムは非常に強力です。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 DN は、クライアントアプリケーションが送信した LDAP 操作で指定されます。たとえば、次のようにします。

```
% ldapmodify -D "cn=TimeSheetApplication" -w secretpwd -y
"cn=manager,dc=siroe,dc=com" -b "siroe.com" -f weekly_update.ldif
```

この `ldapmodify` コマンドでは、アプリケーションがマネージャエントリ (`cn=manager`) の権限を使用して、`weekly_update.ldif` ファイルにある変更を行えることを示しています。マネージャのパスワードは要求されないことに注意してください。

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。無効にされると、ユーザはディレクトリにバインドできず、認証操作は失敗します。

アカウントの無効化は、nsAccountLock 操作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、サーバはバインドを拒否します。

ユーザ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 (role) 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、71 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使用されるかを規定した規則の集まりです。Directory Server で使用できるパスワードポリシーのメカニズムでは、パスワードの長さやユーザによるパスワードの再使用の可否などを指定できます。ユーザがディレクトリへのバインドを試行すると、ディレクトリはそのパスワードとユーザのディレクトリエントリのパスワード属性に格納されている値を比較し、マッチしているかどうかを確認します。また、Directory Server は、ユーザにディレクトリへのバインドを許可する前に、パスワードポリシーで定義されている規則を使用して、パスワードが有効であるかどうかを確認します。

パスワードポリシーの属性

ここでは、サーバのパスワードポリシーを作成するために設定する属性について説明します。次の項目ごとに属性を説明します。

- ユーザ定義のパスワード
- リセット後のパスワードの変更
- パスワードの有効期限
- 期限切れの警告
- パスワードの構文検査
- パスワード長
- パスワードの最短有効日数
- パスワードの履歴

- パスワード保存スキーマ

リセット後のパスワードの変更

Directory Server のパスワードポリシーでは、初回のログイン後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、ユーザのイニシャル、ユーザ 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、初回のログイン後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、ユーザはパスワードを変更するように要求されます (詳細は、138 ページの「ユーザ定義のパスワード」を参照してください)。

パスワードの変更をユーザに許可しないようにした場合、管理者は簡単に推測できる表記規則に従ったパスワードを割り当てるのではなく、簡単に見破られないパスワードを設定します。

デフォルトでは、リセット後にユーザがパスワードを変更する必要はありません。

ユーザ定義のパスワード

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 ID、あるいは、簡単に見破られる可能性があるユーザ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、ユーザ定義のパスワードは許可されています。

パスワードの有効期限

パスワードポリシーでは、ユーザが同じパスワードを無期限に使用できるように設定することができます。また、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もできます。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。その一方で、パスワードが頻繁に期限切れになると、ユーザ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 ～ 90 日で期限切れにします。

パスワードの有効期限をオフにした場合でも、サーバはパスワードの有効期限を記録しています。つまり、パスワードの有効期限のオプションをオンに戻した場合、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、ユーザパスワードは期限切れになりません。

期限切れの警告

一定の期間が過ぎると、ユーザ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。ユーザがサーバにバインドすると、Directory Server によって警告が表示されます。パスワードの有効期限をオンに設定した場合、ユーザ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、デフォルトではユーザ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では、最小パスワード長のガイドラインなど、パスワード文字列についての構文のガイドラインを定義します。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パスワードポリシーで定義したパスワード構文のガイドラインに従っているかどうかが検査されます。また、パスワードの構文検査メカニズムは、パスワードが「安易な」単語かどうかも検査します。この場合、安易な単語とは、ユーザのエントリの `uid`、`cn`、`sn`、`givenName`、`ou`、または `mail` 属性に格納されているような値のことです。

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

Directory Server では、ユーザパスワードの最低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。

デフォルトでは、最小パスワード長は設定されていません。

パスワードの最短有効日数

指定した期間はユーザにパスワードの変更を許可しないように、Directory Server を設定できます。この機能と `passwordHistory` 属性を組み合わせると、ユーザ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 (`passwordMinAge`) 属性を 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 では、2 ～ 24 のパスワードを履歴に格納するように設定したり、パスワードの履歴を無効にしたりすることができるので、ユーザはパスワードを再使用できます。

パスワードポリシーを設定してパスワードの履歴を有効にした場合、ディレクトリには指定した数の古いパスワードが格納されます。ユーザが 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、ディレクトリはそのパスワードを拒否します。この機能で、覚えやすい数個のパスワードをユーザが再使用することはできません。

履歴機能をオフに設定しても、パスワードは履歴に残ります。つまり、パスワードの履歴オプションをオンに戻した場合でも、パスワードの履歴を無効にする前に履歴に格納されていたパスワードをユーザは再使用できません。

デフォルトでは、サーバはパスワードの履歴を保持しません。

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 Directory Server 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)。この暗号化方式がデフォルト

- UNIX 暗号化アルゴリズム

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。暗号化アルゴリズムは、UNIX のパスワードと互換性があります。SSHA がもっとも安全な選択です。

レプリケーション環境でのパスワードポリシーの設計

複製環境では、パスワードポリシーとアカウントロックアウトポリシーが次のように適用されます。

- パスワードポリシーがデータマスター (data master) に適用される
- アカウントロックアウトポリシーは、レプリケーションの対象となるすべてのサーバに適用される

ディレクトリ内にあるパスワードポリシー情報の一部は複製されます。次の属性が複製されます。

- `passwordMinAge` および `passwordMaxAge`
- `passwordExp`
- `passwordWarning`

ただし、設定情報はローカルだけで保持され、複製されません。この情報には、パスワードの構文とパスワードの変更履歴が含まれます。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パスワードポリシーを設定するときは、次の点を考慮します。

- すべての複製は、パスワードの期限切れが近づくと警告を発行する。この情報はローカルの各サーバ上に保持される。したがって、ユーザが複数の複製に順番にバインドした場合、ユーザは同じ警告を数回受信する。また、ユーザがパスワードを変更した場合は、この情報がコンシューマのレプリカで更新されるまで時間がかかることがある。ユーザがパスワードを変更し、その直後にバインドし直した場合は、マスターレプリカに対する変更がコンシューマレプリカに登録されるまではバインドが失敗する場合がある
- マスターや複製を含むすべてのサーバでバインドの動作を一致させたい場合は、各サーバ上でパスワードポリシーの設定情報を一致させる
- 多重マスター環境では、アカウントロックアウトのカウンタが予測できない動作をする場合がある

- 複製のために作成したエントリ (サーバの識別情報など) には、無期限のパスワードを設定する必要がある。これらの特別なユーザ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 (有効範囲の上限の値) を指定する

アカウントのロックアウトポリシーの設計

ディレクトリのパスワードポリシー (password policy) を定義したら、アカウントのロックアウトポリシーを設定して、ユーザのパスワードを潜在的な脅威から保護することができます。

ロックアウトポリシーをパスワードポリシーと組み合わせて使用すると、セキュリティが向上します。パスワードポリシーを設定し、ユーザが一定の回数バインドに失敗したら、そのユーザをディレクトリからロックアウトすることができます。

アカウントのロックアウト機能を使用すると、ハッカーがユーザパスワードを繰り返し推測して、ディレクトリに侵入しようとするのを防止できます。アカウントロックアウトのカウンタは、Directory Server と同じマシン上にあります。この機能はディレクトリサービスからのグローバルなロックアウトとして設計されているわけではありません。これは、レプリケーション環境においても、アカウントロックアウトのカウンタがレプリケーションされないことを意味します。

アクセス制御の設計

ディレクトリのクライアントの識別情報を特定する 1 つ以上の認証スキーマを定義したら、ディレクトリ内に含まれる情報を保護するためにスキーマをどのように使用するか決定する必要があります。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 つ以上の ACL (アクセス制御リスト (access control list)) を使用して指定します。ディレクトリの ACL は、アクセス権限 (読み取り、書き込み、検索など) を許可または拒否し、指定したエントリやその属性と比較する一連のアクセス制御情報 (ACI) 文で構成されています。

ACL を使用すると、以下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、特定のグループに属するすべてのユーザ、およびディレクトリのすべてのユーザ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。

ACI の形式について

セキュリティポリシーを設計する場合、ディレクトリ内での ACI の表現方法を理解していると役立ちます。また、ディレクトリで設定できる権限を理解していることも有用です。ここでは、ACI のメカニズムの概要を簡単に説明します。ACI の形式については、『iPlanet Directory Server 管理者ガイド』を参照してください。

ディレクトリ ACI の一般的な形式は、次のとおりです。

target permission bind_rule

ACI の変数は、次のように定義されます。

- *target*

  ACI が対象とするエントリ (通常はサブツリー)、属性、またはその両方を指定します。*target* は、ACI が適用されるディレクトリの要素を示します。ACI は 1 つのエントリだけを対象にする場合もありますが、複数の属性を対象にすることもできます。また、*target* には LDAP 検索フィルタを含めることもできます。これにより、共通の属性値を持つ多種多様なエントリに権限を設定できます。

- *permission*

  この ACI で設定される実際の権限を示します。*permission* は、指定した *target* に対して読み取りや検索などの特定のタイプのディレクトリアクセスを、ACI が許可または拒否していることを示します。

- *bind_rule*

  権限が適用されるバインド DN またはネットワークの場所を示します。また、バインド規則で LDAP フィルタを指定することもあり、バインド中のクライアントアプリケーションについてフィルタが true であると評価されると、この ACI がクライアントアプリケーションに適用されます。

つまり、ACI は次のように表現されます。

> 「ディレクトリオブジェクトの *target* に対して *bind_rule* が true の場合、*permission* が許可または拒否される」

permission と *bind_rule* はペアで設定し、各ターゲットに対して複数の *permission bind_rule* ペアを設定できます。これにより、特定のターゲットに複数のアクセス制御を効率的に設定できます。たとえば、次のようにします。

target (*permission bind_rule*) (*permission bind_rule*) . . .

たとえば、Babs Jensen としてバインドしているすべてのユーザに Babs Jensen の電話番号への書き込み権限を設定できます。この権限のバインド規則は、「Babs Jensen としてバインドする場合」と記述している部分です。target は Babs Jensen の電話番号で、permission は書き込み権限です。

ターゲット

ディレクトリで作成した各 ACI で、どのエントリをターゲットとするのかを決定する必要があります。ディレクトリの分岐点を示すディレクトリエントリをターゲットとする場合は、その分岐点だけでなくその子エントリすべてが権限の適用範囲に含まれます。ACI がターゲットとするエントリを明示的に指定しない場合は、その ACI 文が含まれているディレクトリエントリが ACI のターゲットとなります。また、ACI のターゲットとなるデフォルトの属性セットは、ターゲットとなったエントリのオブジェクトクラス構造の中で利用可能なあらゆる属性になります。

ACI ごとに、1 つのエントリだけをターゲットにすることも、1 つの LDAP 検索フィルタに一致するエントリだけをターゲットにすることもできます。

エントリをターゲットとする設定以外にも、そのエントリの属性をターゲットに含めることができます。これにより、属性値の一部だけに適用される権限を設定できます。属性のセットをターゲットとするには、ACI によってターゲットに含まれる属性を明示的に指定するか、ターゲットに含めない属性を明示的に指定します。オブジェクトクラス構造で許可される数個の属性を除くすべての属性に権限を設定する場合は、後者の方法を使用してください。

権限

権限を許可あるいは拒否することができます。一般に、147 ページの「アクセスの許可または拒否」で説明している理由から、権限を拒否することは避けてください。

以下の権限を許可または拒否できます。

- Read (読み取り)

  ディレクトリデータを読み取ることができるかどうかを示します。

- Write (書き込み)

  ディレクトリデータを変更または作成できるかどうかを示します。この権限でディレクトリデータを削除することもできますが、エントリ自体を削除することはできません。エントリ全体を削除するには、削除権限を持っている必要があります。

- Search (検索)

ディレクトリデータを検索できるかどうかを示します。読み取り権限では、検索操作の一部としてディレクトリデータが返された場合にそれを閲覧できるという点で、この権限は読み取り権限とは異なります。たとえば、共通名の検索と、部屋番号の読み取りを許可している場合、共通名の検索の一部として部屋番号が返されることはありますが、部屋番号自体は検索できません。これにより、ディレクトリを検索しても特定の部屋にだれがいるのかを知ることはできません。

- Compare (比較)

  比較操作にこのデータを使用できるかどうかを示します。比較には検索機能もありますが、検索操作で実際のディレクトリ情報が返されることはありません。代わりに、比較した値がマッチしたかを示す簡単なブール値だけが返されます。これは、ディレクトリの認証中に `userPassword` 属性値を照合するために使用されます。

- Selfwrite (本人による書き込み)

  グループ管理用だけに使用されます。この権限を使用すると、自分をあるグループに追加したり、削除することができます。Selfwrite はプロキシ認証で使用されます。これは、グループエントリに対して、バインドしたユーザの DN ではなく、プロキシ DN の追加や削除を行うための権限を与えます。

- Add (追加)

  子エントリを作成できるかどうかを示します。この権限を使用すると、対象のエントリの下に子エントリを作成できます。

- Delete (削除)

  エントリを削除できるかどうかを示します。この権限を使用すると、対象のエントリを削除できます。

- Proxy (プロキシ認証)

  ディレクトリマネージャ DN 以外の DN を使用して、その DN の権限でユーザがディレクトリにアクセスできることを示します。

バインド規則

バインド規則は通常、権限が適用されるバインド DN を示します。また、時刻や IP アドレスなどのバインド属性を指定することもできます。

バインド規則を使用すると、ユーザ自身が所有しているエントリだけに ACI が適用されることを簡単に表すことができます。これにより、ユーザが自分以外のエントリを更新することを防ぎ、各ユーザが自分のエントリだけを更新できるように設定できます。

バインド規則を使用すると、次の場合に ACI が適用可能であることを示すことができます。

- バインド操作が特定の IP アドレスや DNS ホスト名から行われている場合のみ。これは、ディレクトリへのすべての更新が、特定のマシンまたはネットワークドメインから行われるように強制する場合によく使用される
- ユーザが匿名でバインドした場合。匿名バインドの権限を設定すると、ディレクトリにバインドするすべてのユーザにその権限が適用されることになる
- ディレクトリに正常にバインドしたすべてのユーザについて。これは、匿名アクセスを防ぐ一方、一般的なアクセスを許可する
- クライアントがそのエントリの直接の親としてバインドした場合のみ
- ユーザがバインドに使用したエントリが、特定の LDAP 検索条件を満たす場合

これらのタイプのアクセスをより簡単に表現するために、次のキーワードを使用できます。

- Parent (親)

  バインド DN が直接の親エントリの場合、バインド規則は true です。たとえば、これにより、あるディレクトリの分岐点ですぐ下の子エントリを管理できるという特定の権限を許可できます。

- Self (本人)

  バインド DN がアクセスを要求しているエントリと同じ場合、バインド規則は true です。たとえば、個々のユーザに自分のエントリを更新できるという特定の権限を許可できます。

- All (すべて)

  ディレクトリに正常にバインドしたすべてのユーザについて、バインド規則は true です。

- Anyone (全員)

  すべてのユーザについて、バインド規則は true です。このキーワードによって、匿名アクセスを許可または拒否します。

権限の設定

デフォルトでは、すべてのユーザに対してすべてのタイプの権限が拒否されています。ディレクトリマネージャは例外です。このため、ユーザがディレクトリにアクセスできるようにするには、ディレクトリにいくつかの ACI を設定する必要があります。

次に、Directory Server が備えているアクセス制御のメカニズムについて説明します。ディレクトリでの ACI の設定方法については、『iPlanet Directory Server 管理者ガイド』を参照してください。

優先規則

ユーザ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、ディレクトリにアクセスするすべてのユーザにこの権限を適用した場合、ユーザ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。特定のユーザにディレクトリ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を除外しておく必要があります。また、対象となるユーザ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。ディレクトリへのアクセスを明示的に拒否する場合は、慎重に行ってください。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

許可特権だけを作成することにより、明示的な拒否特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がある場合は、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、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。

アクセス制御規則を置く場所

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。ユーザの中には、この情報の公開を希望する者もいれば、「非公開」を希望する者もいます。このような場合は、次のようにして対処します。

- 各ユーザのディレクトリエントリに publishHomeContactInfo という属性を作成する
- 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する。LDAP 検索フィルタを使用して、この規則のターゲットを示す
- ディレクトリユーザが自分の publishHomeContactInfo 属性の値を TRUE または FALSE に変更できるようにする。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が決定できる

LDAP 検索フィルタの使用方法と、ACI での LDAP 検索フィルタの使用については、『iPlanet Directory Server 管理者ガイド』を参照してください。

ACI の使用 : ヒントと秘訣

セキュリティポリシーを適用する前に知っておくべきヒントを以下に示します。これらのヒントは、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、ディレクトリの性能向上にも役立ちます。

次のヒントの中には、この章ですでに説明したものも含まれますが、内容を完全なものにするために、もう一度ここでも説明します。

- ディレクトリに含まれる ACI の数を最小限にする

  Directory Server は 50,000 以上の ACI を評価することができますが、多数の ACI 文を管理するのは困難です。ACI の数が多すぎると、特定のクライアントが利用できるディレクトリオブジェクトを即時に判断するのが難しくなります。

  iPlanet Directory Server 5.1 には、マクロを使用することによりディレクトリ内の ACI の数を最小限に抑える新しい機能が備わっています。マクロは、ACI の中で DN、または DN の一部を表現するために使用される可変部分です。ACI のターゲット部分、バインド規則、またはその両方で、DN を表現するためにマクロを使用できます。マクロ ACI については、『iPlanet Directory Server 管理者ガイド』の「アクセス制御の管理」を参照してください。

- 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複数の ACI を使用する必要がなくなります。

- ACI では最小の属性セットを指定する

これは、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が更新できるようにする場合は、それらの特定の属性を除くすべてについて書き込み権限を許可する 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 Console 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

  可能なかぎり、すでに標準ユーザ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、70 ページの「ディレクトリエントリのグループ化」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。たとえば、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。

SSL による接続のセキュリティ保護

ユーザを識別する認証スキーマとディレクトリ内の情報を保護するアクセス制御スキーマの設計が完了したら、サーバとクライアントアプリケーションの間でやり取りされる情報の整合性をどのように確保するのかを計画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 プロトコルを使用できます。

SSL は、RSA 社が開発した RC2 および RC4 暗号化アルゴリズムと組み合わせて使用することができます。特定の接続で使われる暗号化方式は、クライアントアプリケーションと Directory Server の間のネゴシエーションの結果で選択されます。

また SSL は、転送中に情報が変更されていないことを保証するハッシュメカニズムである、CRAM-MD5 と組み合わせて使用することもできます。

Directory Server では、SSL でセキュリティ保護された接続と SSL を使用しない接続を同時に使用することができます。

SSL の使用については、『iPlanet Directory Server 管理者ガイド』を参照してください。

その他のセキュリティ関連資料

安全なディレクトリの設計方法については、以下の資料を参照してください。

- iPlanet Security Developer Central
  http://developer.iplanet.com/tech/security/
- 『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

ディレクトリの設計例

企業の規模や業種によって、ディレクトリの設計方法は異なります。この章では、さまざまに設定を想定したディレクトリの導入例を紹介します。この章の例を利用して、独自のディレクトリの導入計画を立てることができます。

この章では、次のような組織にディレクトリを導入する際の設計例を紹介します。

- 一般企業
- 多国籍企業およびそのエクストラネット

一般企業

自動車部品の製造業者である siroe.com 社は、社員 500 人の小規模な会社です。siroe.com 社は、Directory Server を導入して、現在のディレクトリ対応アプリケーションをサポートすることを決定しました。siroe.com 社のディレクトリの設計過程を、次の設計段階ごとに説明します。

データの設計

siroe.com 社が最初に行うのは、ディレクトリに格納するデータのタイプの特定です。そのために、siroe.com 社は、導入チームを編成して、これからのディレクトリの使用状況を決定するサイト調査を実施します。導入チームは、次の事項を決定しました。

- ディレクトリにユーザとグループの情報を保持し、全社的に導入されている iPlanet サーバベースのイントラネットをサポートする。ユーザとグループの情報のほとんどを、ディレクトリの管理者グループが集中的に管理する。ただし、電子メールの情報は別のメールの管理者グループが管理する
- ディレクトリを、iPlanet Messaging Server、iPlanet Web Server、iPlanet Calendar Server、人事管理アプリケーション、および個人別電話帳アプリケーションで使用する
- iPlanet Messaging Server では、`uid`、`mailServerName`、`mailAddress` などの属性に対して完全一致検索を行う。データベースの性能を向上させるために、siroe.com 社は iPlanet Messaging Server で検索をサポートするこれらの属性のインデックスを管理する

  インデックスの使用については、96 ページの「インデックスを使用したデータベース性能の向上」を参照してください。
- 個人別電話帳アプリケーションでは、ユーザ名と電話番号が頻繁に検索される。そのため、ディレクトリは、大量の検索結果を返す、膨大な数の部分文字列検索、ワイルドカード検索、および近似 (音による) 検索を処理できなければならない。そのため siroe.com 社では、`dn`、`sn`、および `givenName` の各属性に対して近似インデックスと部分文字列インデックスを使用する
- 将来、s/mime 電子メールなどの公開鍵インフラストラクチャ (PKI) アプリケーションをサポートする予定なので、ディレクトリ内にユーザの公開鍵証明書を格納できるように準備する

スキーマの設計

siroe.com 社の導入チームは、ディレクトリ内のエントリを表すために `inetOrgPerson` オブジェクトクラス (object class) を使用することを決定しました。このオブジェクトクラスを使用するのは、siroe.com 社のディレクトリでサポートされるアプリケーションが必要とする `userCertificate` と `uid(userID)` 属性の両方を使用できるためです。

また、siroe.com 社はデフォルトのディレクトリスキーマ (schema) をカスタマイズしたいと考えています。siroe.com 社は、自社の社員を表すために `siroePerson` オブジェクトクラスを作成します。このオブジェクトクラスは、`inetOrgPerson` オブジェクトクラスから派生させます。

siroePerson オブジェクトクラスでは、siroeID 属性だけを使用できるようにします。この属性には、siroe.com 社の各社員に割り当てられている特別な社員番号が含まれます。

siroe.com 社は将来、必要に応じて、新しい属性を siroePerson オブジェクトクラスに追加することができます。

ディレクトリツリーの設計

siroe.com 社は、次のようにディレクトリツリー (directory tree) を作成します。

- ディレクトリツリーのルートを、siroe.com 社のインターネットドメインである dc=siroe,dc=com に設定する
- ディレクトリツリーに、ou=people、ou=groups、ou=roles 、および ou=resources の 4 つの分岐点を持たせる
- 全社員のエントリを、ou=people 分岐の下に作成する

  社員のエントリはすべて、person、organizationalPerson、inetOrgPerson、および siroePerson の各オブジェクトクラスのメンバーになります。uid 属性は、各エントリの DN を一意に識別します。たとえば、siroe.com 社のエントリには Babs Jensen 用のエントリ (uid=bjensen) と Emily Stanton 用のエントリ (uid=estanton) が含まれます。
- 営業、マーケティング、経理の各部署に 1 つずつ、計 3 つのロールを作成する

  各ユーザエントリに、ユーザが所属する部署を特定するロール (role) 属性を追加します。これにより、これらのロールに基づいて ACI を作成できます。ロールについては、71 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。
- ou=groups 分岐の下に 2 つのグループの分岐を作成する

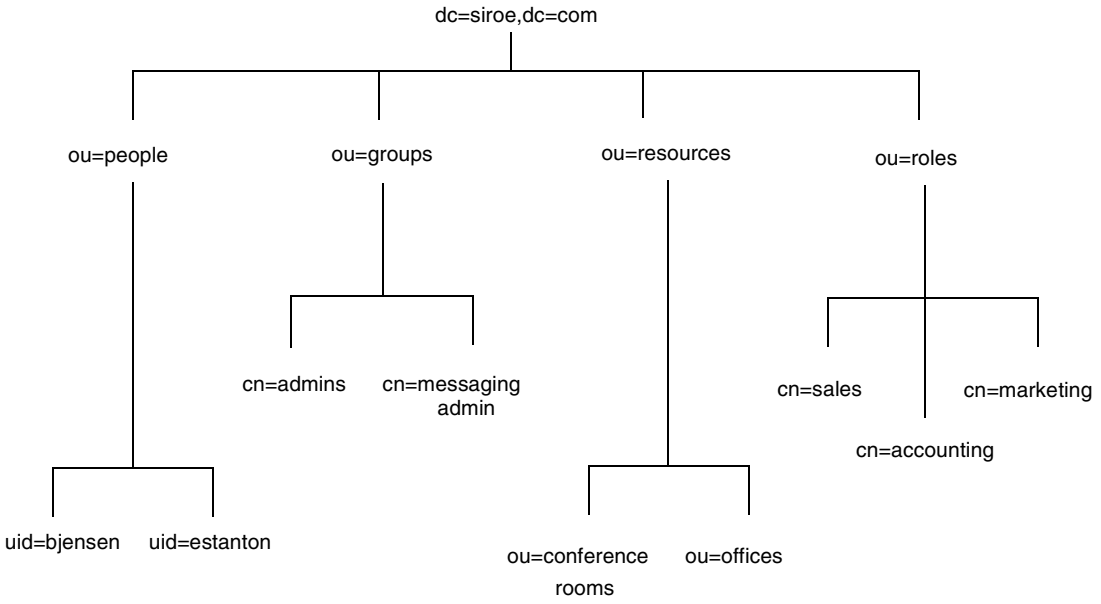
  最初のグループ cn=administrators に、ディレクトリの内容を管理するディレクトリ管理者用のエントリを追加します。

  メール管理者は、2 番目のグループ cn=messaging admin を使用して、メールアカウントを管理します。このグループは、iPlanet Messaging Server で使用する管理者グループと対応させます。siroe.com 社では、Messaging Server 用に構成するグループと Directory Server 用に作成するグループを、別々のグループにします。
- 会議室用 (ou=conference rooms) とオフィス用 (ou=offices) にそれぞれ 1 つずつ、計 2 つの分岐を ou=resources 分岐の下に作成する
- エントリが管理者グループに属するかどうかによって、異なる値を mailquota 属性に提供する CoS (サービスクラス (Class of Service)) を作成する

この CoS によって、管理者には 500M バイトのメールサイズが、一般社員には 100M バイトのメールサイズが割り当てられます。CoS については、73 ページの「サービスクラス」を参照してください。

次の図に、上記の設計ステップの結果として作成されたディレクトリツリーを示します。

図 8-1 siroe.com 社のディレクトリツリー



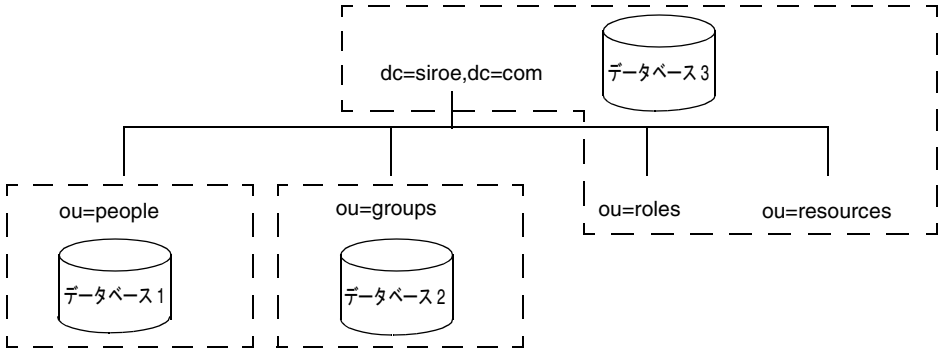
トポロジの設計

次に、siroe.com 社はデータベーストポロジとサーバトポロジの両方を設計します。次に、各トポロジ (topology) について詳しく説明します。

データベーストポロジ

siroe.com 社では、ユーザの分岐を 1 つのデータベース (DB1) に格納し、グループの分岐をもう 1 つのデータベース (DB2) に格納し、資源の分岐、ロールの分岐、およびルート接尾辞 (root suffix) の情報を 3 番目のデータベース (DB3) に格納するように、データベーストポロジを設計します。siroe.com 社のディレクトリのデータベーストポロジは、次のようになります。

図 8-2　siroe.com 社のデータベーストポロジ



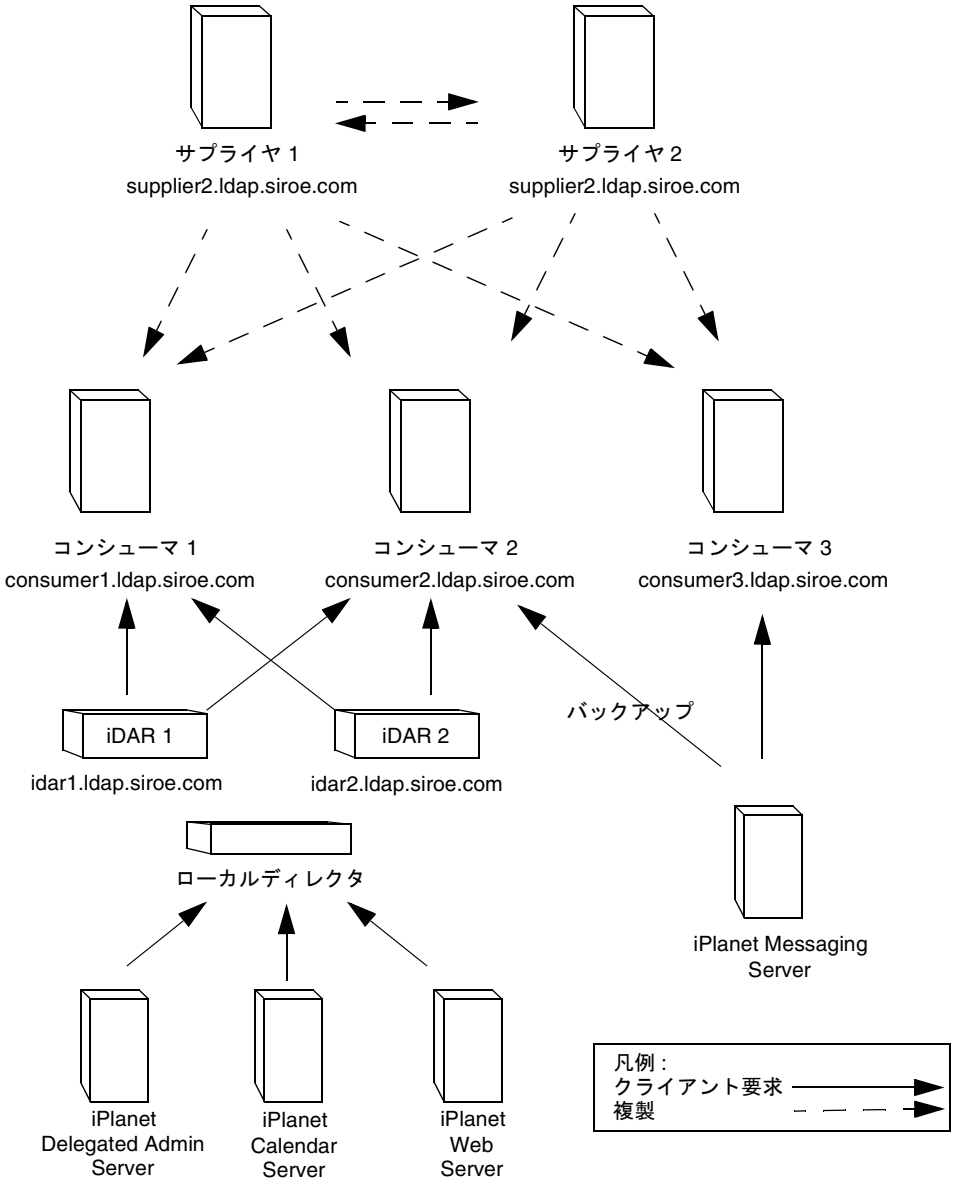
サーバトポロジ

siroe.com 社が導入する Directory Server では、2 つのサプライヤ (supplier) サーバのそれぞれが 3 つのコンシューマ (consumer) サーバのすべてに対して、更新処理を実行します。これらのコンシューマは、iPlanet Data Access Router (iDAR) を使用して、1 つの iPlanet Messaging Server とそれ以外の Unified User Management 製品にデータを提供します。

iPlanet Unified User Management 製品については、
http://www.iplanet.com/products/ を参照してください。

siroe.com 社のサーバトポロジは次のようになります。

図 8-3　siroe.com 社のサーバトポロジ



iPlanet のサーバ製品 (Calendar Server、Delegated Admin Server など) から送られた変更要求は、iDAR によって適切なコンシューマサーバに転送されます。コンシューマサーバはスマートレフェラルを使用して、変更されるデータのマスターコピーを保

持するサプライヤサーバに要求を転送します。

レプリケーションの設計

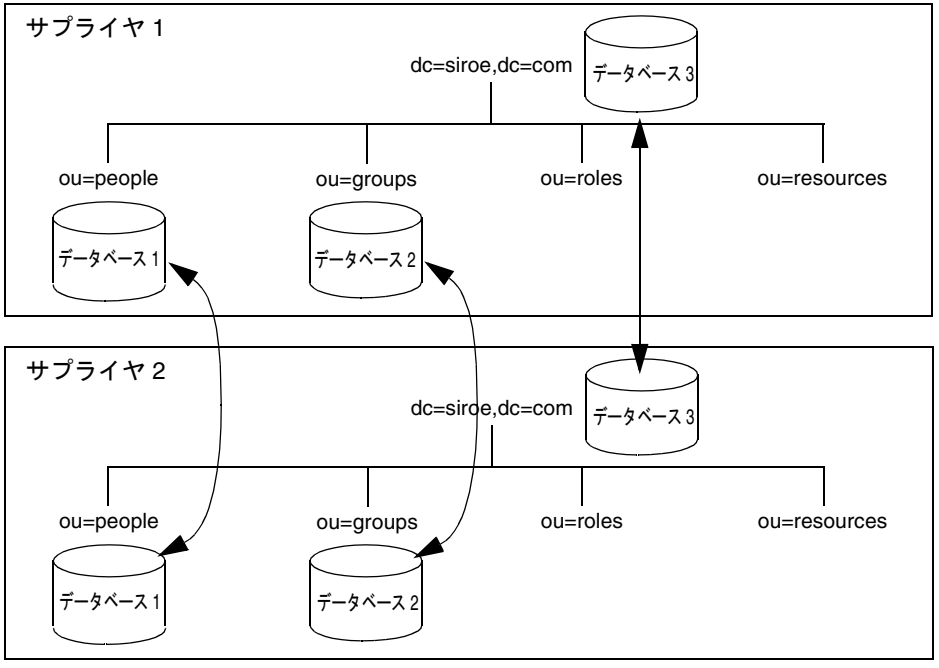
siroe.com 社は、ディレクトリのデータが高い可用性を得るためにマルチマスターレプリケーション (multi-master replication) を使用することにします。マルチマスターレプリケーションについては、108 ページの「マルチマスターレプリケーション」を参照してください。

次に、サプライヤサーバのアーキテクチャとサプライヤサーバ / コンシューマサーバのトポロジについて詳しく説明します。

サプライヤのアーキテクチャ

siroe.com 社では、マルチマスターレプリケーションアーキテクチャで 2 つのサプライヤサーバを使用します。これらのサプライヤは、ディレクトリデータの整合性を維持するために、互いに更新を行います。サプライヤのアーキテクチャを次の図に示します。

図 8-4 siroe.com 社のサプライヤのアーキテクチャ



サプライヤ / コンシューマアーキテクチャ

siroe.com 社が導入するディレクトリで、サプライヤサーバから各コンシューマにレプリケーションが行われる方法を次の図に示します。

図 8-5 siroe.com 社のサプライヤ / コンシューマアーキテクチャ

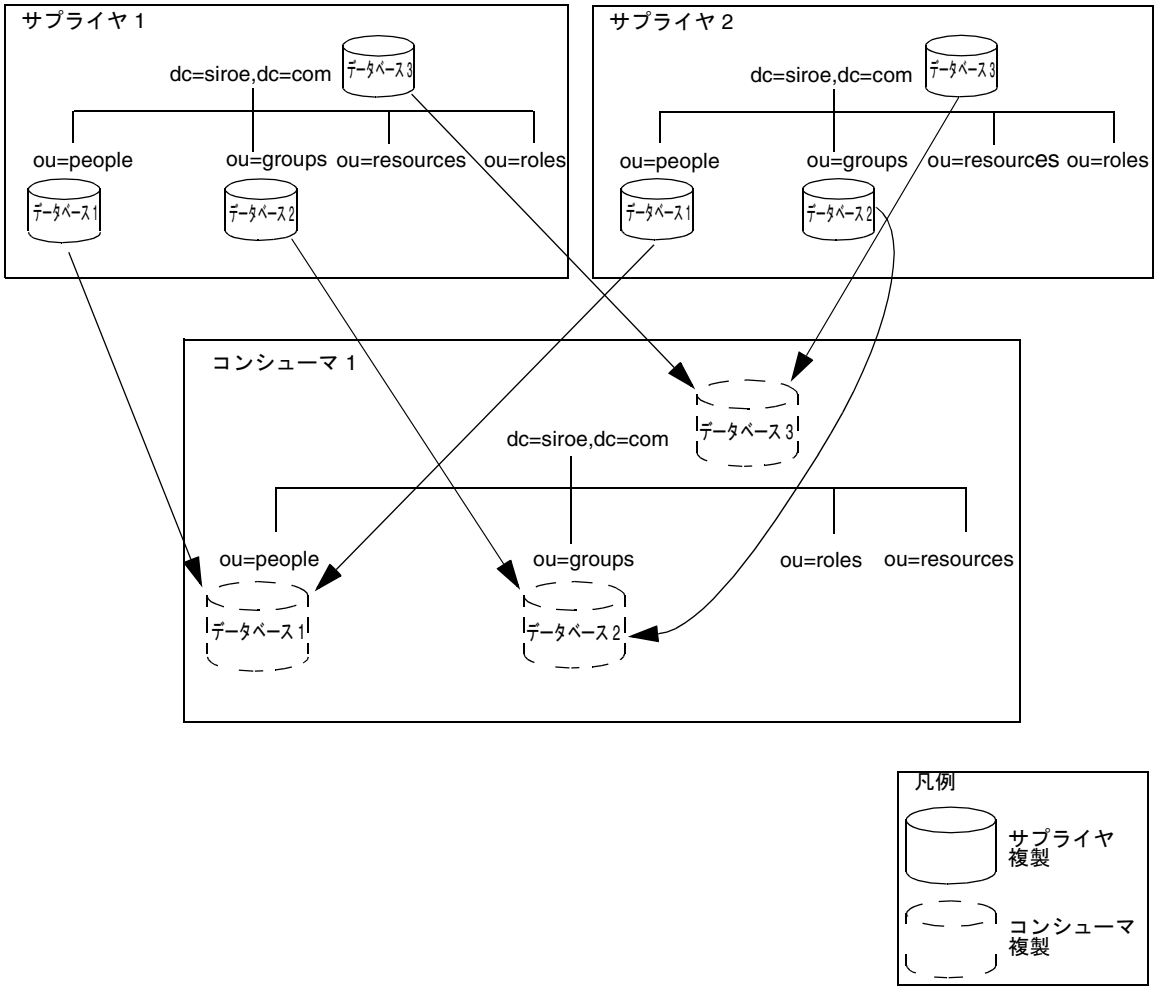


図 8-5 に示すように、2 つのサプライヤサーバによって 3 つのコンシューマサーバがそれぞれ更新されます。これにより、1 つのサプライヤサーバが故障しても、コンシューマサーバは影響を受けずに済みます。

セキュリティの設計

siroe.com 社は、ディレクトリのデータを保護するために次のようなセキュリティの設計を決定しました。

- 社員が自分のエントリを変更できるような ACI を作成する

  ユーザは、uid、manager、および department 属性を除くすべての属性を変更できます。

- 社員データの機密性を保護するために、社員とその上司だけに社員の自宅の住所と電話番号の読み取り権限を与える ACI を作成する

- ディレクトリツリーのルートに、2 つの管理者グループに対して適切なディレクトリへのアクセス権を与える ACI を作成する

  ディレクトリの管理者グループは、ディレクトリへのフルアクセスを必要とします。メッセージングの管理者グループは、mailRecipient オブジェクトクラスと mailGroup オブジェクトクラス、これらのオブジェクトクラスに含まれている属性、および mail 属性への書き込みおよび削除権限を必要とします。また、siroe.com 社は、メッセージングの管理者グループがメールグループを作成できるように、グループのサブディレクトリへの write、delete、および add の各権限を与えます。

- ディレクトリツリーのルートに汎用的なアクセス制御を作成し、匿名アクセスに対して読み取り、検索、および比較の各権限を許可する

  この ACI では、匿名ユーザによるパスワード情報へのアクセスは拒否されます。

- サービス拒否攻撃および不正な使用からサーバを保護するために、ディレクトリクライアントがバインドに使用する DN に基づいて資源制限を設定する

  siroe.com 社では、検索要求の応答として受信するエントリ数を、匿名ユーザに 100 エントリまで、管理者ユーザに 1,000 エントリまで許可し、システム管理者には制限を加えません。ユーザのバインド DN に基づく資源制限の設定方法については、『iPlanet Directory Server 管理者ガイド』の「ユーザアカウントの管理」を参照してください。

- パスワードの長さを 8 文字以上に、パスワードの有効期限を 90 日に設定するパスワードポリシー (password policy) を作成する

  パスワードポリシーについては、137 ページの「パスワードポリシーの設計」を参照してください。

- 経理ロール (role) のメンバーにすべての給与関連情報へのアクセス権を与える ACI を作成する

チューニングと最適化

siroe.com 社では、導入するディレクトリを次の方法で最適化します。

- `idsktune` ユーティリティを実行する (Solaris 9 プラットフォームでは `directoryserver idsktune` を実行する)

  このユーティリティを使用すると、使用システムのパッチレベル、カーネル、およびシステムのネットワーク設定を簡単に確認できます。`idsktune` については、『iPlanet Directory Server インストールガイド』を参照してください。

- エントリとデータベースのキャッシュを最適化する

  siroe.com 社では、すべてのインデックスが確実に RAM に収まるように、エントリキャッシュを 2,000 エントリに設定し、データベースキャッシュを 250M バイトに設定して、サーバの性能を最適化します。

運用に関する決定

siroe.com 社では、ディレクトリの日々の運用に関して次を事項を決定しました。

- データベースを毎晩バックアップし、週に 1 度バックアップをテープに書き込む
- SNMP を使用してサーバの状態を監視する

  SNMP については、『iPlanet Directory Server 管理者ガイド』を参照してください。

- アクセスログを自動的にローテーションさせる
- エラーログを監視して、サーバが正常に動作しているかどうかを確認する
- アクセスログを監視して、インデックス付けが必要な検索を選別する

アクセスログ、エラーログ、および監査ログについては、『iPlanet Directory Server 管理者ガイド』の「サーバとデータベースアクティビティの監視」を参照してください。

多国籍企業およびそのエクストラネット

次に、siroe.com International 社のディレクトリインフラストラクチャを構築する例を示します。前述の例の siroe.com 社が大規模な多国籍企業に成長したと仮定します。ここで示す例では、siroe.com 社で作成したディレクトリ構造を元に、ディレクトリの設計を拡張して新しい要件を満たしていきます。

siroe.com International 社は、主要な 3 つの地域である米国、ヨーロッパ、およびアジアに拠点を持つ組織へと成長しました。社員数は 20,000 人を超えており、社員はそれぞれ siroe.com International 社のオフィスがある国に住み、働いています。siroe.com International 社は、全社規模の LDAP ディレクトリを導入して、社内通信の改善、Web アプリケーションの開発と導入の簡素化、およびセキュリティと機密性の改善を行うことを決定しました。

国際企業向けのディレクトリツリーを設計する場合は、論理的にディレクトリエントリを収集する方法、データ管理をサポートする方法、および世界規模で複製を可能にする方法を決定する必要があります。

また、siroe.com International 社では、部品供給業者および取引先が使用できるエクストラネットも作成します。エクストラネットは、企業のイントラネットを外部のクライアントに対して拡張したものです。

次に、siroe.com International 社が多国籍ディレクトリサービスおよびエクストラネットを導入する過程を設計段階ごとに説明します。

- 163 ページの「データの設計」
- 164 ページの「スキーマの設計」
- 165 ページの「ディレクトリツリーの設計」
- 167 ページの「トポロジの設計」
- 172 ページの「レプリケーションの設計」
- 175 ページの「セキュリティの設計」

データの設計

siroe.com International 社では、導入チームを編成して、サイト調査を実施します。導入チームは、サイト調査によって次の事項を決定しました。

- ほとんどのサイトで、iPlanet Messaging Server を使用して、電子メールの経路指定、配信、および閲覧サービスを提供する。企業サーバを使用して、文書公開サービスを提供する。すべてのサーバを、Solaris UNIX オペレーティングシステム上で稼働させる

- ローカルでもデータ管理ができるようにする。たとえば、ヨーロッパのサイトは、ディレクトリ内のヨーロッパの分岐を管理する責任がある。これは、ヨーロッパのサイトが、そのサイトに保管されているデータのマスターコピーに対しても責任を持つことを意味する
- オフィスが地理的に広い地域に分散しているため、ユーザとアプリケーションからは 1 日に 24 時間ディレクトリを利用できる必要がある
- 多くのデータ要素は、複数の言語と文字セットで表現されるデータ値を格納する必要がある

導入チームは、エクストラネットのデータ設計に関して次の事項を決定しました。

- 供給業者が siroe.com International 社のディレクトリにログインして、siroe.com International 社との契約を管理できるようにする。供給業者は、名前やパスワードなど認証に使用するデータ要素に依存する
- siroe.com International 社の取引先が、ディレクトリを使用して、取引先のネットワークで電子メールアドレスと電話番号を検索できるようにする

スキーマの設計

siroe.com International 社は、エクストラネットをサポートするスキーマ要素を追加することにより、元のスキーマの設計を拡張します。そのため、`siroeSupplier` オブジェクトクラスと `siroePartner` オブジェクトクラスの 2 つの新しいオブジェクトを追加します。

`siroeSupplier` オブジェクトクラスでは、`siroeSupplierID` 属性だけを使用できます。この属性には、siroe.com International 社が取引先である各自動車部品供給業者に割り当てた一意の ID が含まれます。

`siroePartner` オブジェクトクラスでは、`siroePartnerID` 属性だけを使用できます。この属性には、siroe.com International 社が各取引先に割り当てた一意の ID が含まれます。

デフォルトのディレクトリスキーマのカスタマイズ方法については、46 ページの「スキーマのカスタマイズ」を参照してください。

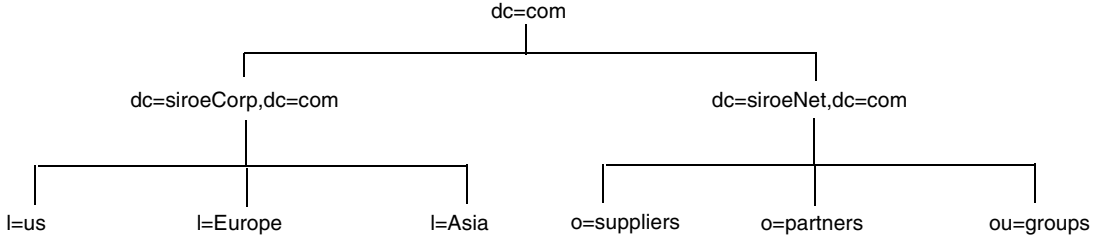
ディレクトリツリーの設計

siroe.com International 社は、次のようにディレクトリツリーを作成します。

- ディレクトリツリーのルート接尾辞を dc=com にする。この接尾辞の下に 2 つの分岐を作成する。分岐の 1 つ dc=siroeCorp,dc=com には、siroe.com International 社の社内データを入れる。もう 1 つの分岐 dc=siroeNet,dc=com には、エクストラネットのデータを入れる
- イントラネットのディレクトリツリー (dc=siroeCorp,dc=com の下) には、3 つの主な分岐を作成する。各分岐は、siroe.com International 社のオフィスがある地域と対応させる。これらの分岐は、l(locality) 属性を使用して識別する
- dc=siroeCorp,dc=com の下の各分岐は、siroe.com 社に元からあったディレクトリツリーの設計と同じにする。siroe.com International 社は、各地域分岐の下に、ou=people、ou=groups、ou=roles、および ou=resources の各分岐を作成する。ディレクトリツリーの設計については、156 ページの「siroe.com 社のディレクトリツリー」を参照
- dc=siroeNet,dc=com 分岐の下に 3 つの分岐を作成する。供給業者用に 1 つの分岐 (o=suppliers)、取引先用に 1 つの分岐 (o=partners)、グループ用に 1 つの分岐 (ou=groups) を作成する
- エクストラネットの ou=groups 分岐には、エクストラネットの管理者用のエントリだけでなく、取引先が参加する、自動車部品の製造に関する最新情報を提供するメーリングリストも入れる

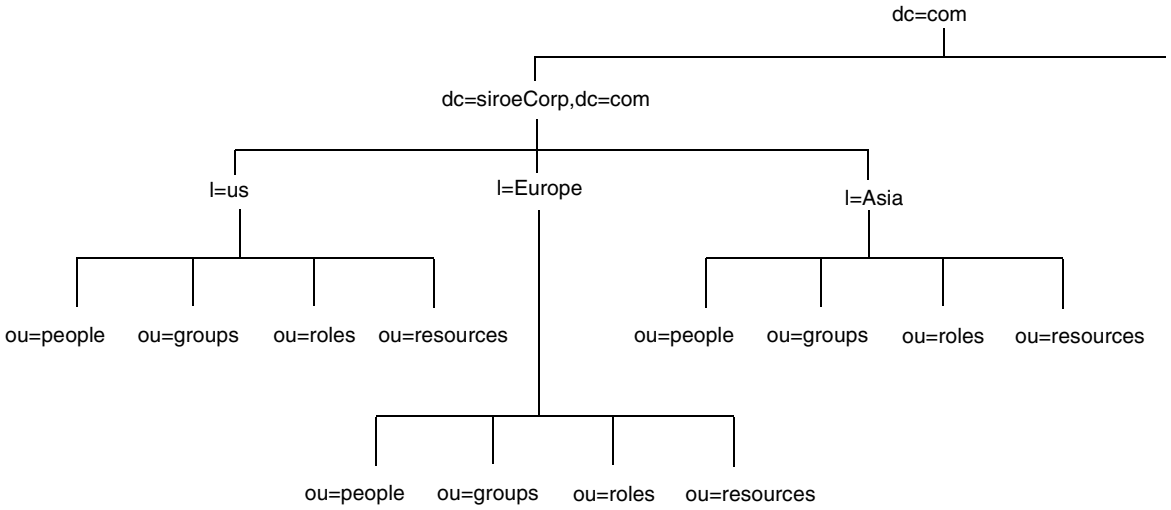
この結果、次のような基本的なディレクトリツリーが作成されます。

図 8-6 siroe.com International 社の基本的なディレクトリツリー



siroe.com International 社のイントラネット用のディレクトリツリーは、次のようになります。

図 8-7　siroe.com International 社のイントラネット用のディレクトリツリー

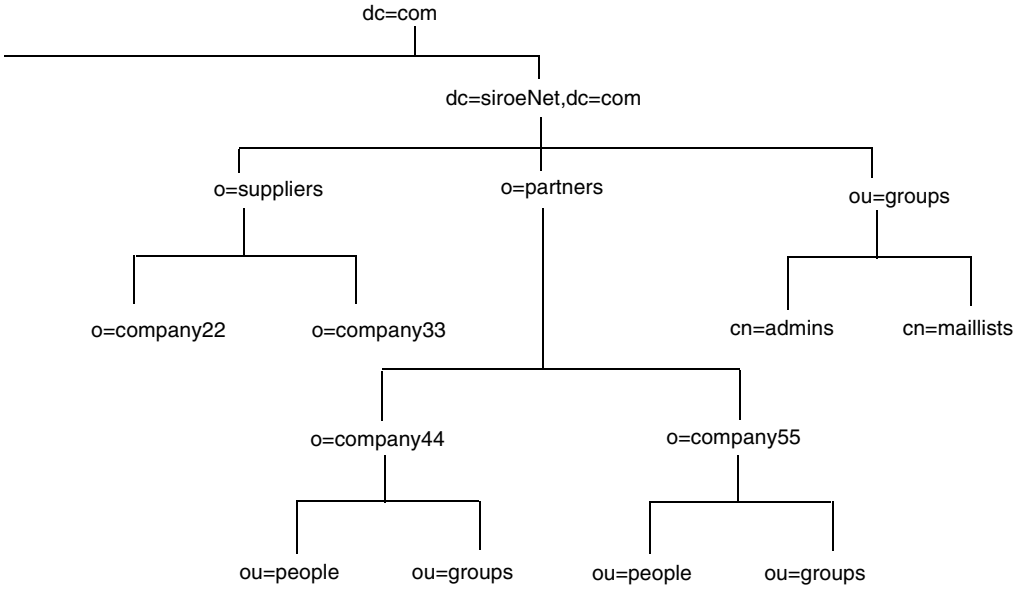


`l=Asia` エントリは、LDIF では次のようなエントリとして表示されます。

```
dn: l=Asia,dc=siroeCorp,dc=com
objectclass: top
objectclass: locality
l: Asia
description: includes all sites in Asia
```

siroe.com International 社のエクストラネット用のディレクトリツリーは、次のようになります。

図 8-8　siroe.com International 社のエクストラネット用のディレクトリツリー



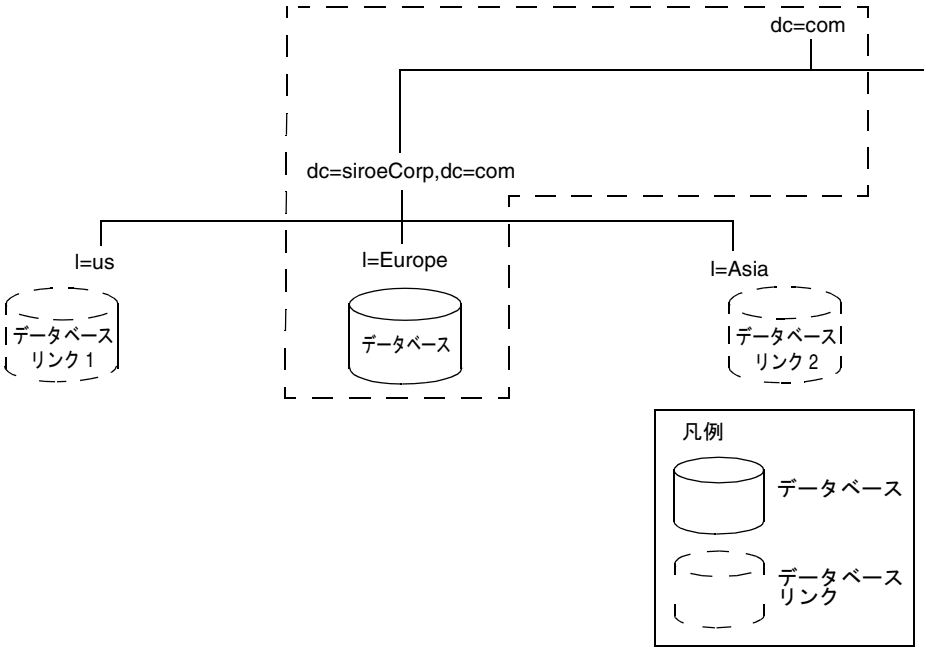
トポロジの設計

次に、siroe.com International 社はデータベーストポロジとサーバトポロジの両方を設計します。以下に、各トポロジについて詳しく説明します。

データベーストポロジ

siroe.com International 社の主要な拠点の 1 つである、ヨーロッパ支社のデータベーストポロジを次の図に示します。

図 8-9　siroe.com International 社のヨーロッパ支社のデータベーストポロジ



データベースリンクは、各国でローカルに格納しているデータベースをポイントします。たとえば、siroe.com International 社のヨーロッパ支社のサーバが受信した、`l=US` 分岐の下にあるデータ対して行われた操作要求は、データベースリンク (database link) によってテキサス州オースティンにあるサーバ上のデータベースに連鎖されます。データベースリンクと連鎖 (chaining) については、91 ページの「連鎖の使用方法」を参照してください。

図 8-9 の点線が示すように、`dc=siroeCorp,dc=com` にあるデータのマスターコピーとルートエントリの `dc=com` は、ヨーロッパで保持されています。

ヨーロッパのデータセンターには、エクストラネットのデータのマスターコピーが保管されています。エクストラネットのデータは、主要な分岐にそれぞれ 1 つずつ配置されている 3 つのデータベースに格納されています。エクストラネットのデータベーストポロジを次の図に示します。

図 8-10　siroe.com International 社のエクストラネット用のデータベーストポロジ

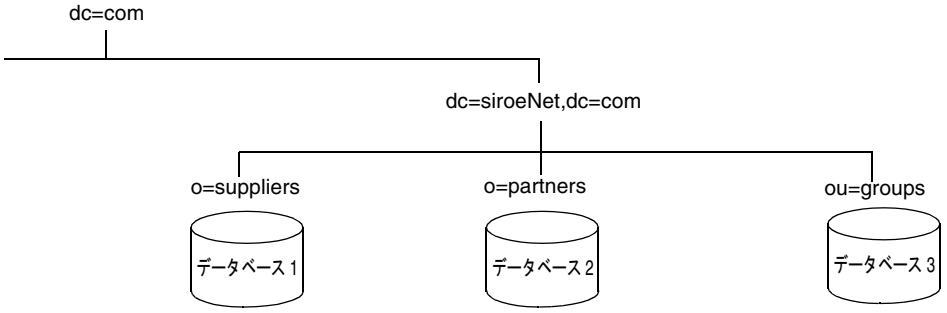


図 8-10 に示すように、o=suppliers にあるデータのマスターコピーはデータベース 1 に、o=partners にあるデータのマスターコピーはデータベース 2 に、ou=groups にあるデータのマスターコピーはデータベース 3 に、それぞれ格納されています。

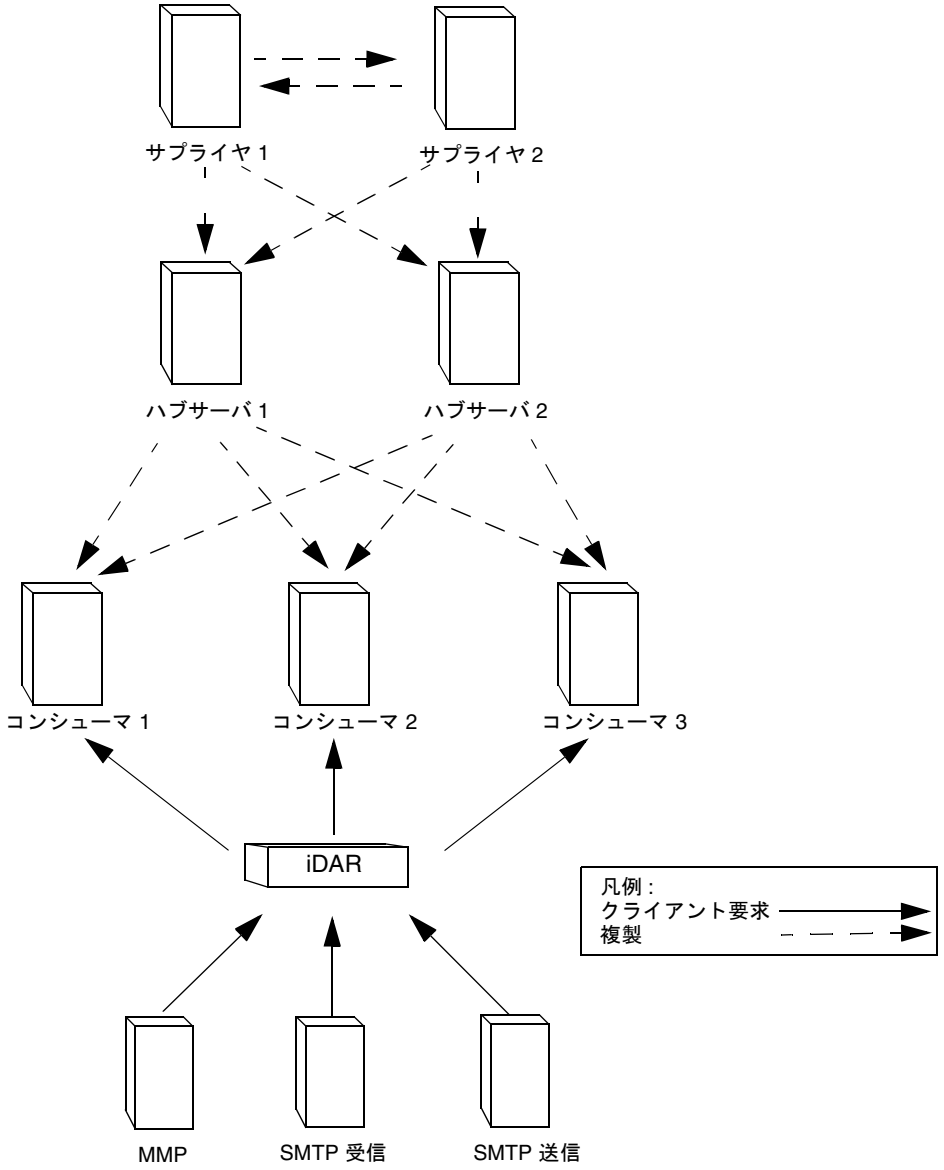
サーバトポロジ

siroe.com International 社では、企業内イントラネット用と、取引先との間で使用するエクストラネット用に、それぞれ 1 つずつ、計 2 つのサーバトポロジを構築します。

siroe.com International 社では、イントラネット用に、主要な拠点がそれぞれマスターデータベースを保持することにします。これはイントラネット用に 3 つのデータセンターが存在することを意味し、各データセンターには 2 つのサプライヤサーバ、2 つのハブサーバ、および 3 つのコンシューマサーバが配置されます。

siroe.com International 社のヨーロッパ支社にあるデータセンターのアーキテクチャは、次のようになります。

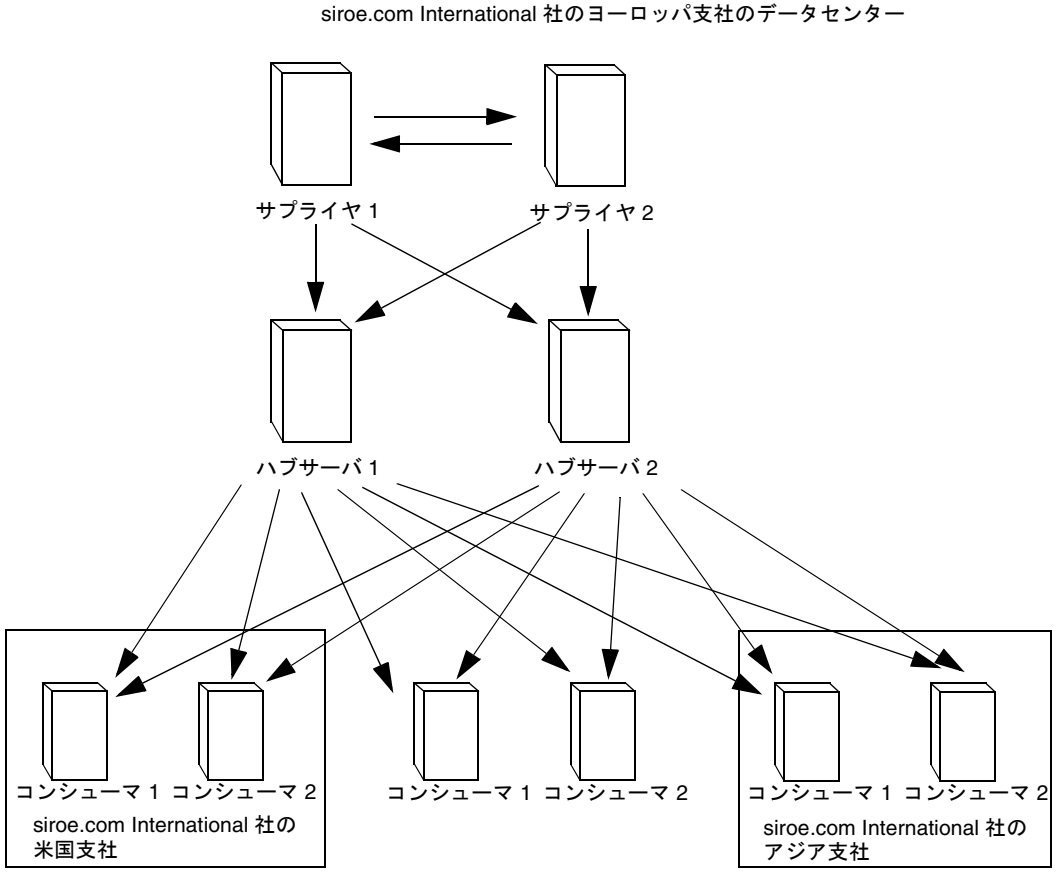
図 8-11　siroe.com International 社のヨーロッパ支社のサーバトポロジ



エクストラネットのデータマスターは、ヨーロッパに置きます。このデータは、米国にあるデータセンターの 2 つのコンシューマサーバと、アジアにあるデータセンターの 2 つのコンシューマサーバに複製されます。つまり、siroe.com International 社ではエクストラネットをサポートするために 10 台のサーバが必要です。

siroe.com International 社のエクストラネット用サーバのアーキテクチャは、ヨーロッパ支社のデータセンターからは次のように見えます。

図 8-12 siroe.com International 社のエクストラネット用のサーバトポロジ



データは、ハブサーバによって、ヨーロッパ支社のデータセンターにある 2 つのコンシューマサーバ、米国支社のデータセンターにある 2 つのコンシューマサーバ、アジア支社のデータセンターにある 2 つのコンシューマサーバにレプリケーションされます。

レプリケーションの設計

siroe.com International 社は、次の事項を考慮してディレクトリのレプリケーションを設計します。

- データはローカルで管理する
- ネットワークの接続の品質は、サイトによって異なる
- データベースリンクを使用して、リモートサーバ上のデータに接続する
- データの読み取り専用コピーが格納されているハブサーバは、データをコンシューマサーバに複製するのに使用する

ハブサーバは、メールサーバや Web サーバなどの重要なディレクトリ対応アプリケーションの近くに配置します。

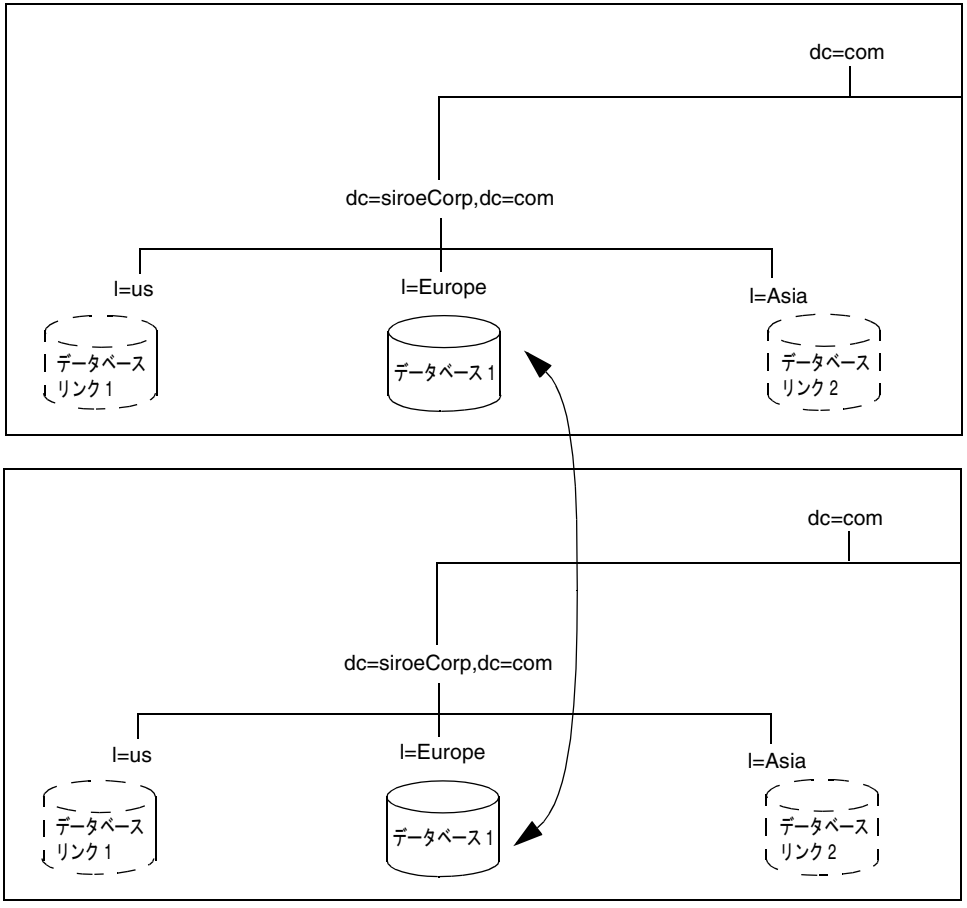
ハブサーバは、サプライヤサーバから複製に伴う負荷を取り除きます。そのため、サプライヤは、書き込み処理に集中できます。将来 siroe.com International 社が発展し、コンシューマサーバを追加する必要が生じても、追加したコンシューマがサプライヤの性能に影響を与えることはありません。

ハブサーバについては、109 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。

サプライヤのアーキテクチャ

siroe.com International 社のイントラネットでは、各拠点で自分が所有するデータのマスターコピーを保持し、データベースリンクを使用してほかの拠点のデータに連鎖しています。各拠点では、所有するデータのマスターコピーについては多重マスター複製アーキテクチャを使用します。たとえば、`dc=siroeCorp,dc=com` と `dc=com` の情報が置かれているヨーロッパのサプライヤのアーキテクチャは次のようになります。

図 8-13　siroe.com International 社のヨーロッパ支社のサプライヤのアーキテクチャ



各拠点には、そのサイトのデータのマスターコピーを共有する 2 つのサプライヤを配置します。したがって、各拠点は自分が所有するデータのマスターコピーに責任を持つことになります。多重マスターアーキテクチャを使用することで、ローカルでのデータの利用が保証され、各サプライヤサーバの管理業務にかかる負荷を均等にすることができます。

ディレクトリ全体が機能しなくなる危険を避けるため、siroe.com International 社では各サイトでマルチマスターディレクトリサーバを使用します。

ヨーロッパにある 2 つのサプライヤサーバと米国にある 2 つのサプライヤサーバとの間の相互動作を次の図に示します。

図 8-14　siroe.com International 社のヨーロッパ支社と米国支社用のサプライヤ対サプライヤのアーキテクチャ

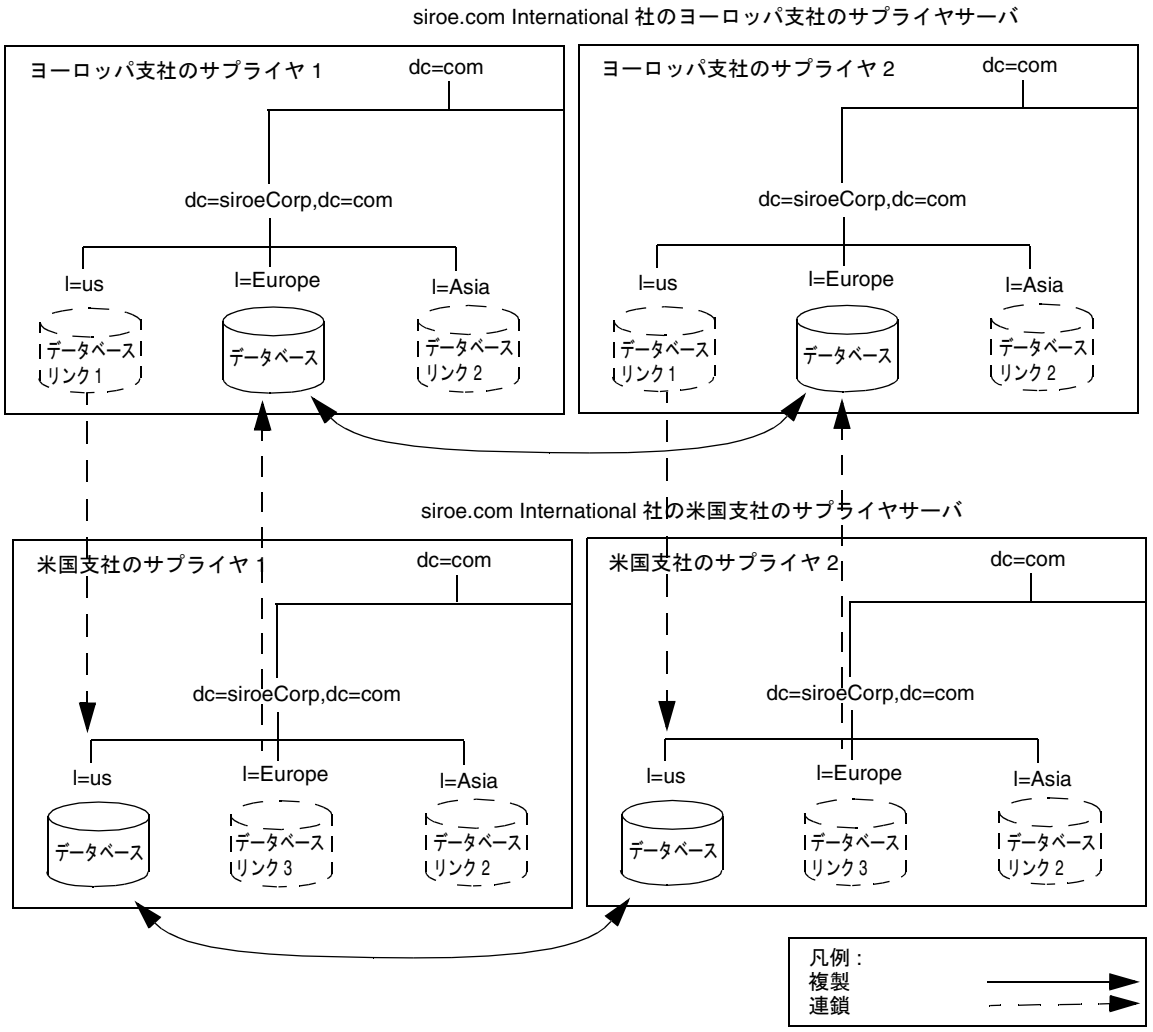


図 8-14 で示したのと同じ関係が、米国支社とアジア支社の間、ヨーロッパ支社とアジア支社の間にも存在します。

セキュリティの設計

siroe.com International 社では、新しい多国籍イントラネットをサポートするために次のアクセス制御を追加して、既存のセキュリティ設計を拡張しています。

- 汎用的な ACI をイントラネットのルートに追加して、制限の厳しい ACI を各国の分岐点とその下の分岐に作成する
- マクロ ACI を使用して、ディレクトリ内の ACI の数を最小限にとどめる

  マクロを使用して、ターゲットの DN および ACI のバインド規則部分を表します。ディレクトリが着信 LDAP 操作を受信すると、その LDAP 操作の対象となる資源と ACI マクロがマッチするかどうかが調べられます。マッチした場合、マクロは対象となる資源の DN の値に置き換えられます。

  マクロ ACI については、『iPlanet Directory Server 管理者ガイド』を参照してください。

siroe.com International 社では、次のアクセス制御を追加してエクストラネットをサポートしています。

- エクストラネットのすべての処理について、証明書に基づく認証を使用することを決定した。ユーザがエクストラネットにログインするときに、デジタル証明書が必要になる。証明書の格納にディレクトリを使用する。ディレクトリに証明書が格納されるので、ユーザはディレクトリに格納されている公開鍵を検索することで暗号化した電子メールを送信できる
- エクストラネットへの匿名アクセスを禁止する ACI を作成する。これにより、サービス拒否攻撃からエクストラネットを保護できる
- siroe.com International 社のディレクトリデータへの更新は、siroe.com International 社がホストしているアプリケーションから送られたもの以外は許可しない。つまり、エクストラネットを使用する取引先と供給業者は、siroe.com International 社が提供するツールしか使用できないようにする。エクストラネットのユーザが使用するツールを siroe.com International 社が許可したものに制限することにより、siroe.com International 社の管理者は監査ログを使用してディレクトリの利用状況を追跡することができ、siroe.com International 社以外のエクストラネットユーザが引き起こす問題を限定することができる
- iDAR を使用してセキュリティを強化する。iDAR については、http://www.iplanet.com/ を参照

用語集

ACI　Access Control Instruction の略称。ディレクトリ内のエントリに対するアクセス権を許可または拒否する命令。

ACL　アクセス制御リスト。ディレクトリへのアクセスを制御するメカニズム。

Authenticating Directory Server　PTA (パススルー認証) における、要求元クライアントの認証資格を保持する Directory Server を指す。PTA が有効なホストは、クライアントから受信する PTA 要求をホストに送信する。

CA　「認証局 (Certificate Authority)」を参照。

ciphertext　この情報を復号化する適切な鍵がないと読むことができない、暗号化された情報。

CIR　「コンシューマ主導レプリケーション (consumer-initiated replication)」を参照。

CoS　サービスクラス。アプリケーションに認識されない方法で、エントリ間で属性を共有する方法。

CoS 定義エントリ (CoS definition entry)　使用中の CoS のタイプを特定する。対象とする分岐の下に LDAP サブエントリとして格納される。

CoS テンプレートエントリ (CoS template entry)　共有属性値のリストを含む。

DAP　Directory Access Protocol の略称。クライアントがディレクトリにアクセスするための ISO X.500 標準プロトコル。

Directory Access Protocol　「DAP」を参照。

Directory Server Console　ディレクトリの内容を表示、設定、および管理するためのグラフィカルユーザインタフェースを提供する LDAP クライアントアプリケーション。iPlanet Directory Server 製品のコンポーネント。

DIT 「ディレクトリツリー (directory tree)」を参照。

DM 「ディレクトリマネージャ (Directory Manager)」を参照。

DN 「識別名 (distinguished name)」を参照。

DNS ドメインネームシステム。標準の IP アドレス (198.93.93.10 など) をホスト名 (`www.iPlanet.com` など) と関連付けるために、ネットワーク上のマシンが使用するシステム。マシンは通常、ホスト名の IP アドレスを DNS サーバから取得するか、システム上で維持されているテーブルから検索する。

DNS エイリアス (DNS alias) DNS サーバが認識しているホスト名で、別のホスト (特に、DNS CNAME レコード) をポイントする。マシンは常に実際の名前を 1 つ持つが、1 つ以上のエイリアスを持つこともできる。たとえば、`www.[yourdomain].[domain]` などのエイリアスは、現在サーバが存在する `realthing.[yourdomain].[domain]` という名前の実際のマシンをポイントできる。

HTML ハイパーテキストマークアップ言語。World Wide Web 上のドキュメントで使用されるフォーマット化言語。HTML ファイルはフォーマット化コードを含むプレーンテキストファイルであり、Netscape Navigator などのブラウザにテキストの表示方法、グラフィックの配置方法、および項目の配列方法を指示し、ほかのページへのリンクを表示する。

HTTP ハイパーテキスト転送プロトコル。HTTP サーバとクライアントの間で情報を交換するための規約。

HTTP-NG 次世代のハイパーテキスト転送プロトコル。

HTTPD HTTP デーモンまたはサービスの略称で、HTTP プロトコルを使用して情報を提供するプログラム。一般に、このデーモンまたはサービスは、httpd と呼ばれる。

HTTPS セキュリティ保護を強化した HTTP。SSL (Secure Sockets Layer) を使用して実装される。

IP アドレス (IP address) インターネットプロトコルアドレス。ドットで区切られた一組の数字で、インターネット上にあるマシンの実際の位置を指定する。たとえば、198.93.93.10 など。

ISO 国際標準化機構。

LDAP Lightweight Directory Access Protocol の略称。TCP/IP を介して複数のプラットフォーム間で動作するように設計されたディレクトリサービスプロトコル。

LDAP Data Interchange Format 「LDIF」を参照。

LDAP URL　DNS を使用して Directory Server を検出し、LDAP を介して照会を完了する方法を提供する。たとえば、`ldap://ldap.iplanet.com` など。

LDAP クライアント (LDAP client)　LDAP Directory Server からの LDAP エントリを要求および表示するために使用されるソフトウェア。「ブラウザ (browser)」も参照。

LDAPv3　LDAP プロトコルのバージョン 3。Directory Server のスキーマ形式は、このプロトコルに基づく。

LDBM データベース (LDBM database)　高性能なディスクベースのデータベースで、このデータベースに割り当てられたすべてのデータを含む一連の大きなファイルで構成される。Directory Server の一次データ記憶域である。

LDIF　LDAP Data Interchange Format の略称。Directory Server のエントリをテキスト形式で表すために使用される形式。

Lightweight Directory Access Protocol　「LDAP」を参照。

MD5　RSA Data Security, Inc. によるメッセージダイジェストアルゴリズム。データの短いダイジェストの生成に使用できる。このダイジェストは、高い確率で一意となるため、同じメッセージダイジェストを生成するデータの作成は、数学的に見て非常に困難である。

MD5 シグニチャ　MD5 アルゴリズムで生成されたメッセージダイジェスト。

MIB　管理情報ベース。SNMP ネットワークと関連付けられたすべてのデータ、またはその一部。MIB は、すべての SNMP 管理対象オブジェクトの定義を含むデータベースとみなすことができる。MIB は、ツリーに似た階層を持つ。最上位にはネットワークに関するもっとも一般的な情報が含まれており、下位では個別のネットワーク領域に固有の情報を扱う。

MIB ネームスペース (MIB namespace)　管理情報ネームスペース。ディレクトリのデータに名前を設定し、参照する方法。ディレクトリツリーとも呼ばれる。

N + 1 ディレクトリ問題 (n + 1 directory problem)　さまざまなディレクトリで同じ情報の複数のインスタンスを管理する場合の問題。結果的に、ハードウェアにかかる費用と人的費用が増大する。

Network Management Station　「NMS」を参照。

NIS　Network Information Service の略称。UNIX マシンが制御する、プログラムとデータファイルから構成されるシステムで、コンピュータのネットワーク全体のマシン、ユーザ、ファイルシステム、およびネットワークパラメタに関する各マシン固有の情報を収集、照合、および共有するためのサービスを提供する。

NMS　Network Management Station の略称。1 つ以上のネットワーク管理アプリケーションがインストールされたパワフルなワークステーション。

ns-slapd　iPlanet LDAP Directory Server のデーモンまたはサービスで、Directory Server のすべてのアクションに関連する。「slapd」も参照。

OID　「オブジェクト識別子 (object identifier)」を参照。

PDU　Protocol Data Unit の略称。SNMP デバイス間のデータ交換の基礎となる符号化されたメッセージ。

Protocol Data Unit　「PDU」を参照。

PTA　パススルー認証。バインド資格を確認するために、1 つの Directory Server がほかの Directory Server と交信するメカニズム。

PTA Directory Server　パススルー認証 (PTA) で、受信したバインド要求を Authenticating Directory Server に送信 (パススルー) するサーバ。

PTA LDAP URL　パススルー認証で、Authenticating Directory Server、パススルーサブツリー、および省略可能なパラメタを定義する URL。

RAM　ランダムアクセスメモリ。コンピュータ内部にあり、多数の半導体で構成された物理的な記憶装置。RAM 内に格納されている情報は、コンピュータが停止すると消失する。

rc.local　マシンの起動時に実行されるプログラムを記述した Unix マシン上のファイル。格納位置から、`/etc/rc.local` とも呼ばれる。

RDN　相対識別名。完全な識別名を形成するために文字列にエントリの祖先を追加する前の、エントリ自体の名前。

RFC　Request For Comments の略称。インターネットコミュニティに提出される手順あるいは標準文書。技術が標準として受け入れられる前に、ユーザは技術に関してコメントを送ることができる。

root　Unix マシン上でもっとも高いレベルの特権を持つユーザ。root ユーザは、マシン上のすべてのファイルに対して完全なアクセス特権を持つ。

Secure Sockets Layer　「SSL」を参照。

SIE　サーバインスタンスのエントリ。

SIR　「サプライヤ主導レプリケーション (supplier-initiated replication)」を参照。

slapd　LDAP Directory Server のデーモンまたはサービス。複製以外のディレクトリのほとんどの機能を受け持つ。「ns-slapd」も参照。

SNMP　簡易ネットワーク管理プロトコル。ネットワーク処理に関するデータを交換することによって、サーバ上で実行しているアプリケーションプロセスを監視および管理するために使用される。

SNMP サブエージェント (SNMP subagent)　管理対象の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すソフトウェア。

SNMP マスターエージェント (SNMP master agent)　さまざまなサブエージェントと NMS の間で情報を交換するソフトウェア。

SSL　Secure Sockets Layer の略称。クライアントとサーバとの間にセキュリティ保護された接続を確立するソフトウェアライブラリ。セキュリティ保護が強化された HTTP である HTTPS の実装に使用される。

TCP/IP　Transmission Control Protocol/Internet Protocol の略称。インターネットや企業内ネットワークにおける主要なネットワークプロトコル。

TLS　Transport Layer Security の略称。SSL の新標準で、公開鍵に基づいたプロトコル。

Transport Layer Security　「TLS」を参照。

uid　Unix システム上で、各ユーザと関連付けられた一意の番号。

URL　Uniform Resource Locator の略称。サーバおよびクライアントが文書の要求に使用するアドレス指定システム。ロケーションとも呼ばれる。URL の形式は、`[protocol]://[machine:port]/[document]`。ポート番号は一部のサーバでのみ必要であり、多くの場合サーバによって割り当てられるので、その場合ユーザは URL でポート番号を指定する必要はない。

X.500 標準 (X.500 standard)　Directory Server の実装で使用される、推奨する情報モデル、オブジェクトクラス、および属性を概説する一連の ISO/ITU-T 文書。

アカウントの無効化 (account inactivation)　ユーザアカウント、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する。

アクセス権 (permission)　アクセス制御で、ディレクトリ情報へのアクセスの許可または拒否、および許可または拒否されるアクセスのレベルを規定する。「アクセス権限」も参照。

アクセス権限 (access rights)　アクセス制御で、許可または拒否されているアクセスのレベルを指す。アクセス権限は、ディレクトリで実行できる操作のタイプと関連している。読み取り、書き込み、追加、削除、検索、比較、本人による書き込み、プロキシなど、すべての権利を許可または拒否できる。

アクセス制御命令 (access control instruction)　「ACI」を参照。

アクセス制御リスト (access control list) 「ACL」を参照。

入れ子状のロール (nested role) ほかのロールを含むロールの作成が可能。

インデックスキー (index key) ディレクトリが使用する各インデックスは、インデックスキーのテーブルとマッチングエントリ ID リストで構成されている。

エントリ (entry) オブジェクトに関する情報を含む LDIF ファイル内の行のグループ。

エントリ ID リスト (entry ID list) ディレクトリが使用する各インデックスは、インデックスキーのテーブルとマッチングエントリ ID リストで構成されている。エントリ ID リストは、クライアントアプリケーションの検索要求とマッチする可能性があるエントリ候補のリストを構築するために、ディレクトリが使用する。

エントリの分散 (entry distribution) 多数のエントリをサポートできるようスケーリングするために、複数のサーバにディレクトリエントリを配布する手法。

オブジェクトクラス (object class) どの属性がそのエントリ内に含まれるのかを定義することにより、ディレクトリ内のエントリのタイプを定義する。

オブジェクト識別子 (object identifier) オブジェクト指向システムにおいて、オブジェクトクラスや属性などのスキーマ要素を一意に特定する、通常 10 進数の数字の文字列。オブジェクト識別子は、ANSI、IETF、または同様の組織が割り当てる。

親アクセス (parent access) この権限が与えられると、バインド DN がアクセス先エントリの親である場合は、ユーザはディレクトリツリー内で自分の下にあるエントリにアクセスできる。

カスケード型レプリケーション (cascading replication) カスケード型レプリケーションでは、1 つのサーバ (一般にハブサプライヤと呼ばれる) が特定の複製でコンシューマとサプライヤの両方として動作する。このサーバは読み取り専用の複製を保持し、更新履歴ログを管理する。また、データのマスターコピーを保持するサプライヤサーバから更新を受け取り、次にコンシューマにこの更新を供給する。

仮想リスト表示インデックス (virtual list view index) ブラウズインデックスとも呼ばれる。Directory Server Console でエントリ内の表示を高速化する。仮想リスト表示インデックスは、表示の性能を向上させるために、ディレクトリツリー内のすべての分岐点で作成可能。

簡易ネットワーク管理プロトコル (Simple Network Management Protocol) 「SNMP」を参照。

間接 CoS (indirect CoS) 間接 CoS は、ターゲットエントリの属性のうちの 1 つの値を使用してテンプレートエントリを特定する。

管理されているロール (managed role) ユーザは、メンバーの明示的な列挙リストを作成できる。

管理情報ベース (management information base) 「MIB」を参照。

管理対象オブジェクト (managed object) SNMP エージェントがアクセス可能で、NMS に対しても送信できる標準値。各管理対象オブジェクトは、ドット表記法で表現される正式名および数字の識別子で識別される。

近似インデックス (approximate index) 類似 (音による) の用語を探すのに有効な近似検索を許可する。

クライアント (client) 「LDAP クライアント (LDAP client)」を参照。

クラシック CoS (classic CoS) DN およびターゲットエントリの属性値の 1 つを使用して、テンプレートエントリを特定する。

クラス定義 (class definition) 特定のオブジェクトのインスタンスを作成するために必要な情報を指定し、ディレクトリ内のほかのオブジェクトに関連してそのオブジェクトがどのように動作するのかを決定する。

コードページ (code page) 国際化プラグインでロケールが使用する内部テーブル。オペレーティングシステムが、キーボードのキーを画面に表示するための文字フォントと関連付けるときに使用する。

更新履歴ログ (change log) 複製に対する変更を記述した記録。サプライヤサーバは、コンシューマサーバに格納されているレプリカに対して、またはマルチマスターのレプリケーションの場合はほかのマスターに対して、これらの変更を適用する。

国際標準化機構 (International Standards Organization) 「ISO」を参照。

コンシューマ (consumer) サプライヤサーバからレプリケーションされたディレクトリツリーまたはサブツリーを含むサーバ。

コンシューマサーバ (consumer server) レプリケーションで、ほかのサーバからコピーしたレプリカを保持するサーバは、そのレプリカのコンシューマと呼ばれる。

コンシューマ主導レプリケーション (consumer-initiated replication) コンシューマ (consumer) サーバがサプライヤサーバからディレクトリのデータを引き出すレプリケーション設定。

コンシューマレプリカ (consumer replica) すべての追加および変更操作についてマスターレプリカを参照するレプリカ。サーバは任意の数のコンシューマレプリカを保持できる。

サーバサービス (server service)　実行されると、クライアントからの要求を待機し、受け入れる Windows NT 上のプロセス。Windows NT 上の SMB サーバがこれに当たる。

サーバセレクタ (Server Selector)　ユーザがブラウザを使用してサーバを選択および設定できるインタフェース。

サーバデーモン (server daemon)　実行されると、クライアントからの要求を待機し、受け入れるプロセス。

サーバルート (server root)　サーバのプログラム、設定、管理、および情報のファイルの保持専用の、サーバマシン上のディレクトリ。

サービス (service)　Windows NT マシン上のバックグラウンドプロセスで、特定のシステムタスクを受け持つ。サービスプロセスは、動作を続けるためにユーザの介入を必要としない。

サービスクラス (Class of Service)　「CoS」を参照。

最下位のエントリ (leaf entry)　その下にほかのエントリが 1 つもないエントリ。最下位のエントリは、ディレクトリツリーで分岐点になることはできない。

サブエージェント (subagent)　「SNMP サブエージェント (SNMP subagent)」を参照。

サブ接尾辞 (sub suffix)　ルート接尾辞の下の分岐。

サプライヤ (supplier)　コンシューマサーバに複製されるディレクトリツリーあるいはサブツリーのマスターコピーを保持するサーバ。

サプライヤサーバ (supplier server)　レプリケーションで、別のサーバにコピーされるレプリカを保持するサーバは、そのレプリカのサプライヤと呼ばれる。

サプライヤ主導レプリケーション (supplier-initiated replication)　サプライヤ (supplier) サーバがコンシューマサーバにディレクトリのデータをレプリケーションするレプリケーション設定。

サプライヤレプリカ (supplier replica)　ディレクトリ情報のマスターコピーを含む、更新可能な複製。サーバは任意の数のマスターレプリカを保持できる。

参照整合性 (referential integrity)　関連するエントリ間の関係が、ディレクトリ内で管理されることを保証するメカニズム。

識別名 (distinguished name)　エントリの名前と LDAP ディレクトリ内での位置を文字列で表したもの。

自己アクセス (self access)　この権限が与えられると、バインド DN がターゲットエントリとマッチしている場合は、ユーザは自分のエントリにアクセスできる。

時刻 / 日付の形式 (time / date format)　特定の地域における時刻および日付の習慣的な形式を示す。

システムインデックス (system index)　Directory Server の操作に必須なので削除および変更はできない。

実在インデックス (presence index)　特定のインデックス化された属性を含むエントリの検索を可能にする。

照合順序 (collation order)　ある言語の文字のソート方法について、言語および文化に固有の情報を提供する。この情報には、その文字体系における文字の順序、あるいはアクセント付きの文字とアクセントのない文字とを比較する方法などが含まれる。

証明書 (certificate)　ネットワークユーザの公開鍵を、ディレクトリ内にあるそれらの DN と関連付けるデータの集合。証明書は、ユーザオブジェクトの属性としてディレクトリ内部に格納される。

スーパーユーザ (superuser)　Unix マシン上でもっとも高いレベルの特権を持つユーザ。root とも呼ばれる。スーパーユーザは、マシン上のすべてのファイルに対して完全なアクセス権を持つ。

スキーマ (schema)　ディレクトリにどのようなタイプの情報をエントリとして格納できるかについての定義。スキーマとマッチしない情報がディレクトリに格納されている場合は、そのディレクトリにアクセスを試みているクライアントが正しい結果を表示できないことがある。

スキーマ検査 (schema checking)　ディレクトリ内で追加または変更されたエントリが、定義したスキーマに従っていることを確認する。スキーマ検査はデフォルトでオンになっている。したがって、スキーマに従っていないエントリを格納しようとした場合、エラーメッセージが表示される。

すべての ID のしきい値 (All IDs Threshold)　サーバが管理するすべてのインデックスキーに広域的に適用されるサイズ制限。個々の ID リストのサイズがこの制限値に達すると、サーバによってその ID リストがすべての ID のトークンと置き換えられる。

すべての ID のトークン (All IDs token)　すべてのディレクトリエントリがインデックスキーとマッチするサーバに想定させるメカニズム。実際には、すべての ID のトークンによって、サーバは検索要求で利用可能なインデックスが存在しないかのように動作する。

接尾辞 (suffix)　ディレクトリツリーの頂点にあるエントリの名前で、この下にデータが格納される。同じディレクトリ内に複数の接尾辞が存在できる。各データベースは接尾辞を 1 つだけ持つ。

操作属性 (operational attribute)　操作属性は、ディレクトリが変更およびサブツリーのプロパティを追跡するために内部で使用する情報を含む。明示的に要求しないかぎり、操作属性は検索に応答して返されることはない。

相対識別名 (Relative distinguished name) 「RDN」を参照。

属性 (attribute) エントリを説明する情報を保持する。属性にはラベルと値がある。また、各属性は、属性値として格納される情報のタイプに応じた標準の構文に従う。

属性リスト (attribute list) 特定のエントリタイプまたはオブジェクトクラスに対応する、必須の属性と省略可能な属性のリスト。

ターゲット (target) アクセス制御で、ターゲットは特定の ACI が適用されるディレクトリ情報を識別する。

ターゲットエントリ (target entry) CoS の適用範囲内のエントリ。

対称暗号化 (symmetric encryption) 暗号化と復号化の両方で同じキーを使用する暗号化。対称暗号化アルゴリズムの一例として DES が挙げられる。

国際化インデックス (international index) 国際化情報を含むディレクトリで、検索にかかる時間を短縮する。

マルチマスターレプリケーション (multi-master replication) 2 つのサーバがそれぞれ同じ読み書き可能な複製のコピーを保持する高度なレプリケーションモデル。各サーバは、レプリカの更新履歴ログを保持する。一方のサーバに対する変更は、自動的にもう一方のサーバにもレプリケーションされる。変更が競合した場合、タイムスタンプを使用してどちらのサーバが最新の変更を保持しているかを決定する。

単一マスター複製 (single-master replication) コンシューマサーバに対して 2 つのサーバがそれぞれ同じ読み書き可能なレプリカのコピーを保持する、もっとも基本的なレプリカモデル。単一マスター複製モデルでは、サプライヤサーバが更新履歴ログを管理する。

知識参照 (knowledge reference) さまざまな データベースに格納されているディレクトリ情報へのポインタ。

通貨形式 (monetary format) 特定の地域で使用されている通貨記号や、通貨記号が数値の前と後ろのどちらに付くのか、および通貨単位の表記方法を指定する。

データベースリンク (database link) 連鎖を実装したもの。データベースリンクはデータベースのように動作するが、持続的な記憶領域を持たない。代わりに、リモートに格納されているデータを指し示す。

データマスター (data master) 特定データ部分のマスターソースであるサーバ。

デーモン (daemon) 特定のシステムタスクを担当する、Unix マシン上のバックグラウンドプロセス。デーモンプロセスは、動作の継続に人の介入を必要としない。

定義エントリ (definition entry) 「CoS 定義エントリ (CoS definition entry)」を参照。

ディレクトリサービス (directory service)　組織内の人材および資源に関する記述的な属性ベースの情報を管理するように設計されたデータベースアプリケーション。

ディレクトリツリー (directory tree)　ディレクトリに格納されている情報の論理表現。多くのファイルシステムで使用されているツリーモデルに酷似しており、ツリーのルート点が階層の頂点にある。DIT とも呼ばれる。

ディレクトリマネージャ (Directory Manager)　UNIX の root ユーザに相当する、特権を持ったデータベース管理者。ディレクトリマネージャにはアクセス制御が適用されない。

デフォルトインデックス (default index)　データベースインスタンスごとに作成されるデフォルトインデックスセットの 1 つ。デフォルトインデックスは変更できるが、デフォルトインデックスに依存しているプラグインもあるので、削除する場合は注意が必要。

テンプレートエントリ (template entry)　「CoS テンプレートエントリ (CoS template entry)」を参照。

等価インデックス (equality index)　特定の属性値を含むエントリを効果的に検索できる。

匿名アクセス (anonymous access)　この権限が与えられると、どのユーザも、資格の有無およびバインドの条件とは無関係に、ディレクトリ情報にアクセスできる。

トポロジ (topology)　ディレクトリツリーが複数の物理的なサーバにわたって、どのように分割されているのか、およびこれらのサーバがどのように相互にリンクをしているのかを示す。

名前の衝突 (name collisions)　同じ識別名を持った複数のエントリ。

認証 (authentication)　(1) クライアントユーザの ID を Directory Server に対して示すプロセス。ユーザがディレクトリへのアクセスを許可されるには、バインド DN、および対応するパスワードまたは証明書のどちらかを提示する必要がある。ディレクトリ管理者がユーザに許可したアクセス権に基づき、Directory Server はユーザに機能の実行やファイルおよびディレクトリへのアクセスを許可する。

(2) ほかのコンピュータがそのサーバであるかのように偽装したり、あるいはセキュリティ保護されていないコンピュータにもかかわらず保護されているように装うことを防ぎ、クライアント (client) がセキュリティ保護されたサーバに接続されていることを保証する。

認証局 (Certificate Authority)　認証証明書を販売および発行する会社または組織。ユーザは、信頼する認証局から認証証明書を購入できる。CA とも呼ばれる。

認証証明書 (authentication certificate)　置き換えや偽造の不可能な、第三者が発行するデジタルファイル。認証証明書は、他方を検証し認証するために、サーバからクライアントへ、あるいはクライアントからサーバへ送信される。

ネットワーク管理アプリケーション (network management application) 稼働または停止しているデバイス、受信したエラーメッセージやその数など、SNMP 管理対象のデバイスに関する情報をグラフィカルに表示する Network Management Station コンポーネント。

バインド DN (bind DN) 操作を実行するときに、Directory Server に対する認証で使用される識別名。

バインド規則 (bind rule) アクセス制御で、ディレクトリ情報にアクセスするために特定のユーザまたはクライアントが満たす必要がある資格および条件を指定する。

バインド識別名 (bind distinguished name) 「バインド DN (bind DN)」を参照。

パススルーサブツリー (pass-through subtree) パススルー認証では、PTA Directory Server は、バインド要求をこのサブツリーに DN が含まれているすべてのクライアントから Authenticating Directory Server に渡す (パススルー)。

パススルー認証 (Pass-through authentication) 「PTA」を参照。

パスワードファイル (password file) Unix ユーザのログイン名、パスワード、およびユーザ ID 番号が格納されている Unix マシン上のファイル。格納場所から、`/etc/passwd` とも呼ばれる。

パスワードポリシー (password policy) ディレクトリ内でのパスワードの使い方の基準となる規則のセット。

ハブサプライヤ (hub supplier) レプリケーションで、ほかのサーバからコピーされたレプリカを保持するサーバのことで、このレプリカを第三のサーバにレプリケーションする。「カスケード型複製」も参照。

汎用アクセス (general access) この権限が与えられた場合、認証されたすべてのユーザがディレクトリの情報にアクセスできることを示す。

標準インデックス (standard index) デフォルトで維持されるインデックス。

ファイル拡張子 (file extension) ファイル名のドット (.) より後ろの部分で、通常ファイルタイプを定義する。たとえば、.GIF、.HTML など。`index.html` というファイル名の場合、ファイル拡張子は `html` である。

ファイルタイプ (file type) 特定のファイルの形式。たとえば、グラフィックファイルは GIF 形式で格納される場合が多く、テキストファイルは通常 ASCII テキスト形式で格納される。ファイルタイプは、通常ファイル拡張子 (.GIF、.HTML など) で識別される。

フィルタ (filter) ディレクトリの照会に適用される制約で、返される情報を制限する。

フィルタを適用したロール (filtered role) 各エントリに含まれる属性に応じて、エントリをロールに割り当てることができるようにする。この操作を行うには、LDAP フィルタを指定する必要がある。フィルタにマッチするエントリは、そのロールを所有すると言われる。

レプリカ (replica) レプリカに関与するデータベース。「コンシューマレプリカ (consumer replica)」および「サプライヤレプリカ (supplier replica)」も参照。

レプリケーションアグリーメント (replication agreement) サプライヤサーバに格納されている設定パラメタのセット。複製対象のデータベース、データをプッシュする先のコンシューマサーバ、複製を実行できる時間、コンシューマにバインドするためにサプライヤが使用する DN と資格、および接続をセキュリティ保護する方法を特定する。

レプリケーション (replication) ディレクトリツリーまたはサブツリーをサプライヤサーバからコンシューマサーバにコピーする処理。

部分文字列インデックス (substring index) エントリ内の部分文字列の効率的な検索を可能にする。部分文字列インデックスとして、各エントリの 2 文字以上を指定する必要がある。

ブラウザ (browser) HTML ファイルとして格納されている World Wide Web コンテンツを要求および表示する、Netscape Navigator などのソフトウェア。ブラウザは、ホストサーバとの通信に HTTP プロトコルを使用する。

ブラウズインデックス (browsing index) 仮想表示インデックスとも呼ばれる。Directory Server Console でエントリの表示を高速化する。ディレクトリの性能を向上させるために、ディレクトリツリーのすべての分岐点で作成可能。

プロキシ DN (proxy DN) プロキシ認証で使用される。プロキシ DN とは、クライアントアプリケーションが操作を実行しようとしている対象へのアクセス権を持つエントリの DN。

プロキシ認証 (proxy authorization) 特殊な形式の認証で、ユーザは自分の ID でディレクトリにバインドするが、別のユーザのアクセス権限を付与される。その別のユーザのことをプロキシユーザ、その DN をプロキシ DN と呼ぶ。

プロトコル (protocol) ネットワーク上のデバイスが情報を交換する方法を記述した規則のセット。

分岐エントリ (branch entry) ディレクトリ内でサブツリーの頂点を表すエントリ。

ベース DN (base DN) ベース識別名。検索処理はベース DN に対して行われる。ベース DN とは、ディレクトリツリー内でエントリおよびその下にあるすべてのエントリの DN のこと。

ベース識別名 (Base distinguished name) 「ベース DN (base DN)」を参照。

ポインタ CoS (pointer CoS)　ポインタ CoS は、テンプレート DN だけを使用してテンプレートエントリを識別する。

ホスト名 (hostname)　machine.domain.dom のような書式のマシン名で、IP アドレスに変換される。たとえば、`www.iPlanet.com` は、com ドメインの iPlanet サブドメインにある www というマシンである。

マスターエージェント (master agent)　「SNMP マスターエージェント (SNMP master agent)」を参照。

マッチング規則 (matching rule)　検索処理中にサーバが文字列をどのように比較するかを定めるガイドライン。多言語検索では、サーバが使用する必要がある照合順序および演算子をマッチング規則で規定する。

マッピングツリー (mapping tree)　接尾辞 (サブツリー) の名前をデータベースと関連付けるデータ構造。

マルチプレクサ (multiplexor)　データベースリンクを含むサーバで、リモートサーバと通信する。

文字タイプ (character type)　英字を数字やほかの文字と識別し、また大文字と小文字のマッピングを識別する。

ルート接尾辞 (root suffix)　1 つ以上のサブ接尾辞の親。ディレクトリツリーは複数のルート接尾辞を含むことができる。

レフェラル (referral)　(1) サーバが自身では処理できない検索要求あるいは更新要求を LDAP クライアントから受信すると、サーバは通常、その要求を処理できる LDAP サーバへのポインタをクライアントに返信する。

(2) レプリケーションで、コンシューマレプリカが更新要求を受信すると、対応するマスターレプリカを保持するサーバにこの要求を転送する。この転送プロセスをレフェラルと呼ぶ。

連鎖 (chaining)　要求をほかのサーバに中継するための手法。要求の結果は収集、コンパイルされてから、クライアントに返される。

ロール (role)　エントリをグループ化するメカニズム。各ロールは、そのロールを所有するエントリであるメンバーを持つ。

ロールに基づく属性 (role-based attributes)　関連付けられた CoS テンプレート内にエントリが特定のロールを所有しているため、エントリに記述される属性。

ロケール (locale)　住む地域や、文化、習慣の異なるユーザが、データを表すために使用するもので、照合順序、文字タイプ、通貨形式、時刻 / 日付の形式を識別する。ロケールには、特定言語のデータの解釈方法、格納方法、または照合方法に関する情報が含まれる。また、特定言語を表現するために使用するコードページを提供する。

索引

S

T

U

あ

い

え

と

な

に

ね

は

ひ

ふ

ほ

ま

む

ゆ