Red Hat Directory Server 11

部署指南

用于规划有效目录服务的概念和配置选项

Last Updated: 2023-10-26

Red Hat Directory Server 11 部署指南

用于规划有效目录服务的概念和配置选项

Marc Muehlfeld
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

法律通告

摘要

本指南用于规划目录服务。

目录

使开源更具 INCSIVE

红帽致力于替换我们的代码、文档和 Web 属性中有问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 “CTO Chris Wright” 的信息。

前言

红帽目录服务器部署指南 为规划有效目录服务提供了基础。此处提供的信息适用于设计人员和管理员。

1. 目录服务器概述

Red Hat Directory Server 提供以下关键功能：

- 多supplier 复制 - 为读写操作提供高度可用的目录服务。多层复制可与简单级联复制结合使用，以提供高度灵活且可扩展的复制环境。
- 链和引用 - 通过在单个服务器上存储目录的完整逻辑视图，并透明地在大量目录服务器上维护数据，从而增加该目录的威力。
- 角色和服务类 - 提供灵活的机制，用于在条目之间动态分组和共享属性。
- 有效的访问控制机制 - 提供对宏的支持，大大减少了目录中使用的访问控制声明数量，并提高访问控制评估的可扩展性。
- 通过绑定 DN 来实现资源限制 - 根据客户端的绑定 DN 控制分配给搜索操作的服务器资源量。
- 多数据库 - 提供划分目录数据的简单方法，以简化在目录服务中的复制和锁实施。
- 密码策略和帐户锁定 - 定义一组规则，规定了管理目录服务器中如何管理密码和用户帐户的规则。
- TLS 通过网络提供安全身份验证和通信，使用 Mozilla Network Security Services (NSS) 库进行加密。

Directory 服务器的主要组件包括：

- LDAP 服务器 - LDAP v3 兼容网络守护进程。
- Web 控制台 - 图形化管理控制台，用于减少设置和维护目录服务的努力。
- SNMP 代理 - 可使用简单网络管理协议(SNMP)监控目录服务器。

第 1 章 目录服务简介

红帽目录服务器为内部网、网络和外部网信息提供集中目录服务。目录服务器与现有系统集成并充当整合员工、客户、供应商和合作伙伴信息的集中存储库。目录服务器甚至可以扩展，以管理用户配置文件、首选项和身份验证。

本章介绍了基本想法和概念，以了解目录服务的作用，帮助开始设计目录服务。

1.1. 关于目录服务

术语 *目录服务 (directory service)* 指的是存储与企业、订阅者相关信息的软件、硬件和进程的集合，并为用户提供这些信息。目录服务至少包含一个目录服务器和一个目录客户端程序。客户端程序可以访问存储在目录服务中的名称、电话号码、地址和其他数据。

目录服务的一个示例是域名系统 (DNS) 服务器。DNS 服务器将计算机主机名映射到 IP 地址。因此，所有计算资源（主机）成为 DNS 服务器的客户端。映射主机名允许用户通过记住主机名而不是 IP 地址在网络中轻松定位计算机。DNS 服务器的局限性是它仅存储两种信息：名称和 IP 地址。true 目录服务存储虚拟类型的信息。

目录服务器将所有用户和网络信息存储在单一网络可访问的软件仓库中。很多不同信息可以保存在目录服务器中：

- 物理设备（如组织内打印机）的信息，，如位置、彩色还是黑白、制造商、购买日期和序列号。
- 公共员工信息，如姓名、电子邮件地址和部门。
- 员工的个人信息，如工资、政府身份证明的标识号、主页地址、电话号码和工资级别。
- 合同或帐户信息，如客户端名称、最终交付日期、投标信息、合同号和项目日期。

目录服务器满足各种应用程序的需求。它还提供了一个标准协议和应用程序编程接口(API)，用于访问它所包含的信息。

1.1.1. 关于全局目录服务

目录服务器提供全局目录服务，这意味着它为各种应用程序提供信息。目录服务器是一个管理相同信息的单一解决方案来，而不是尝试将不同的专用数据库捆绑到不同的应用程序（这会造成大量管理负担）。

例如，某个公司运行三个不同的专用电子邮件系统，每个系统都有自己的专有目录服务。如果用户在一个目录中更改密码，则更改不会自动复制到其他目录中。管理相同信息的多个实例会导致硬件和人员成本增加；这个增加的维护成本被成为 *n+1 目录问题*。

全局目录服务通过提供任何应用程序可访问的目录信息的单一集中存储库来解决 n+1 目录问题。但是，为各种应用程序提供对目录服务的访问需要基于网络的方法在应用程序和目录服务之间进行通信。目录服务器使用 LDAP 供应用程序访问其全局目录服务。

1.1.2. 关于 LDAP

LDAP 提供了一个常见语言，供客户端应用和服务器用于相互通信。LDAP 是 ISO X.500 标准描述的目录访问协议 (DAP) 的一种"轻量级"版本。DAP 可让任何应用程序通过可扩展的、可靠的信息框架访问该目录，但具有较高的管理成本。DAP 使用不是互联网标准协议的通信层，并具有复杂的目录修改惯例。

LDAP 保留 DAP 的最佳功能，同时降低管理成本。LDAP 使用通过 TCP/IP 和简化的编码方法运行的开放目录访问协议。它保留数据模型，可在中型投资的硬件和网络基础架构中支持数百万条目。

1.2. DIRECTORY 服务器简介

红帽目录服务器由多个组件组成。目录本身的核心是实施 LDAP 协议的服务器。Red Hat Directory Server 在 LDAP 服务器上有一个客户端图形用户界面，它允许最终用户搜索和更改目录中的条目。其他 LDAP 客户端（第三方程序和使用 Mozilla LDAP SDK 和 OpenLDAP SDK 编写的程序）可与 Red Hat Directory Server 一起使用，或者将其他应用程序与 Red Hat Directory Server 集成。

安装 Red Hat Directory Server 时，它有以下元素：

- 核心目录服务器 LDAP 服务器、LDAP v3 兼容网络守护进程(**ns-slapd**)以及所有相关插件、用于管理服务器及其数据库的命令行工具，及其配置和模式文件。有关命令行工具的更多信息，请参阅 *红帽目录服务器配置、命令和文件参考*。

- 管理服务器，一种控制访问 LDAP 服务器的不同门户的 Web 服务器。有关管理服务器的更多信息，请参阅 *红帽目录服务器管理指南*。

- Web 控制台是一种图形化管理控制台，用于减少设置和维护目录服务的努力。有关 Web 控制台的更多信息，请参阅 *红帽目录服务器管理指南*。

- SNMP 代理使用简单网络管理协议 (SNMP) 监控目录服务器。有关 SNMP 监控的更多信息，请参阅 *红帽目录服务器管理指南*。

在不添加其他 LDAP 客户端程序的情况下，目录服务器可以为 Intranet 或 extranet 提供基础。兼容服务器应用程序使用 目录作为共享服务器信息的中央存储库，如员工、客户、供应商和合作伙伴数据。

目录服务器可以管理用户身份验证、创建访问控制、设置用户首选项以及集中用户管理。在托管环境中，合作伙伴、客户和供应商可以管理自己部分的目录，从而降低管理成本。

1.2.1. 服务器前端概述

目录服务器是一个多线程应用程序。这意味着多个客户端可以同时绑定到同一网络的服务器。随着目录服务的增加，包括大量条目或地理上无数的客户端，它们还包括多个目录服务器位于网络的战略位置。

目录服务器的服务器前端管理与目录客户端程序的通信。多个客户端程序可以通过 TCP/IP（互联网流量协议）和 LDAP 通过 Unix 套接字 (LDAPI) 使用 LDAP 与服务器通信。目录服务器可以与 TLS 建立 *安全*（加密）连接，具体取决于客户端是否协商对传输层安全性(TLS)的使用。

当通信与 TLS 进行时，通信通常会加密。如果客户端已签发证书，则 Directory Server 可以使用 TLS 来确认客户端具有访问服务器的权限。TLS 用于执行其他安全活动，如消息完整性检查、数字签名和服务器之间的相互身份验证。

注意

目录服务器作为守护进程运行；进程是 **ns-slapd**。

1.2.2. 服务器端概述

目录服务器依赖于 *插件* 来为核心服务器添加功能。例如，数据库层是一个插件。目录服务器具有复制、链数据库和其他不同目录功能的插件。

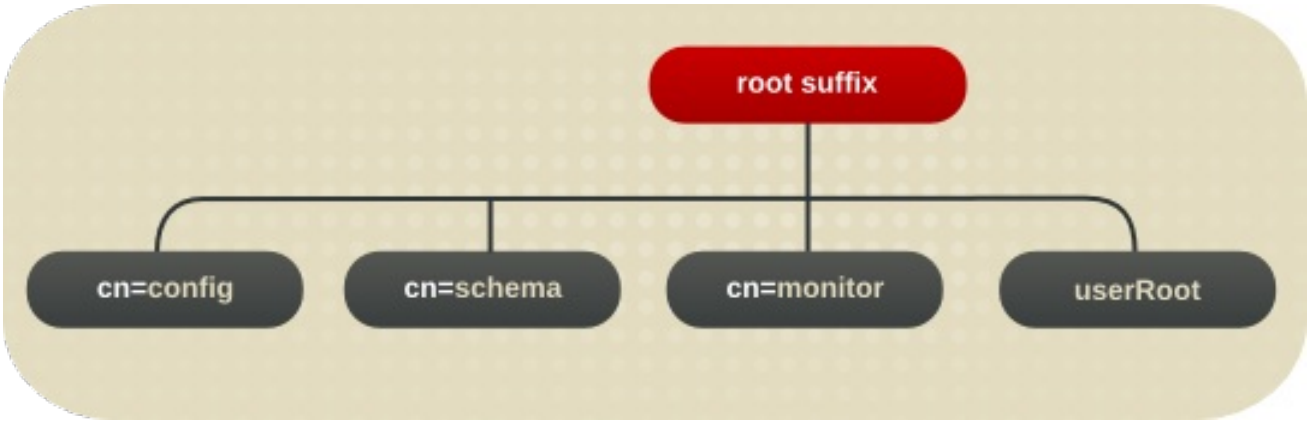
通常，可以禁用插件，特别是扩展服务器功能的插件。禁用后，插件的配置信息将保留在目录中，但其功能不会被服务器使用。根据目录应该执行的操作，可以启用 Directory Server 提供的任何插件来扩展目录服务器功能。（与核心目录服务操作相关的插件，如后端数据库插件，因此无法禁用。）

有关使用 Directory 服务器的默认插件以及编写自定义 *插件的功能的更多信息，请参阅 Red Hat Directory Server 插件指南*。

1.2.3. 基本目录树概述

目录树也称为目录信息树 (DIT)，镜像大多数文件系统使用的树模型，其树结构根或第一个条目出现在层次结构的顶部。在安装过程中，Directory 服务器会创建一个默认目录树。

图 1.1. 默认目录服务器目录树的布局



树的根称为*根后缀*。有关命名根后缀的详情，请参考 第 4.2.1 节 “选择 Suffix”。

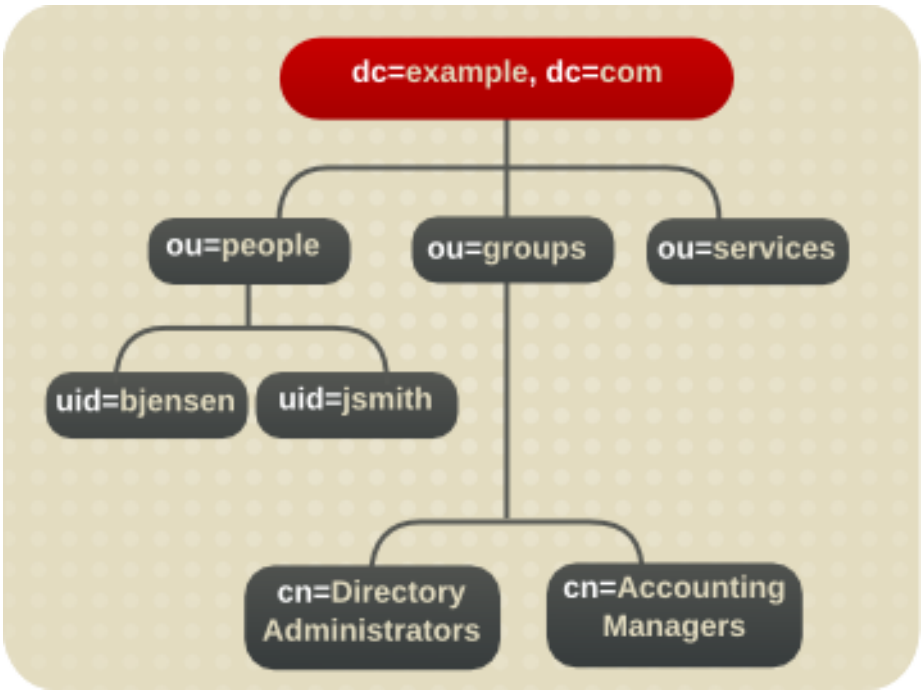
在标准安装后，目录包含根后缀下的三个子树：

- **cn=config**，包含服务器内部配置信息的子树。
- **cn=monitor**，包含目录服务器服务器和数据库监控统计信息的子树。
- **cn=schema**，包含服务器中当前载入的 schema 元素的子树。
- *user_suffix*，设置 Directory 服务器时创建的默认用户数据库的后缀。后缀的名称在服务器创建时由用户定义；关联的数据库的名称为 **userRoot**。可以通过设置时导入 LDIF 文件或条目来填充该数据库。

  user_suffix 后缀经常具有 ***dc*** 命名约定，如 **dc=example,dc=com**。另一个常见的 naming 属性是 ***o*** 属性，用于整个组织，如 **o=example.com**。

可以扩展默认目录树来添加与目录安装相关的任何数据。有关目录树的更多信息，请参阅 第 4 章 *设计目录树*。

图 1.2. 扩展 Example Corp 的 Directory 树.



1.3. 目录服务器数据存储

数据库 是存储、性能、复制和索引的基本单元。所有目录服务器操作 - 导入、导出、备份、恢复和索引条目 - 在数据库上执行。目录数据存储在 LDBM 数据库中。LDBM 数据库作为插件实施，该插件由 目录自动安装并默认启用。

默认情况下，Directory 服务器将一个后端数据库实例用于根后缀 A 单一数据库足以包含目录树。此数据库可以管理数百万条目。

这个数据库支持高级备份和恢复数据的方法，以便最大程度降低数据的风险。

多个数据库可用于支持整个目录服务器部署。信息分布在数据库中，使服务器能够保存比单个数据库更多的数据。

1.3.1. 关于目录条目

LDAP Data Interchange Format(LDIF) 是用于描述目录条目的标准文本格式。*条目* 由 LDIF 文件中的多个行组成（也称为 *段*），它包含对象的相关信息，如机构的人员或网络上的打印机。

有关条目的信息由一组属性及其值在 LDIF 文件中表示。每个条目都有一个对象类属性，用于指定条目描述的对象类型，并且定义它所包含的额外属性集合。每个属性描述一个条目的特定特征。

例如，一个条目可能是对象类 **organizationalPerson**，表示该条目代表一个机构中的人员。这个对象类支持 ***givenname*** 和 ***telephoneNumber*** 属性。分配给这些属性的值提供了条目所代表的人的名称和电话号码。

目录服务器也使用由服务器计算的只读属性。这些属性称为 *操作属性*。管理员可以手动设置可用于访问控制和其他服务器功能的操作属性。

1.3.1.1. 对目录尝试执行查询

条目以分级结构存储在目录树中。LDAP 支持在目录树中查询数据库的工具，并将其低于该条目。此子树的根称为 *基础可分辨名称* 或基本 DN。例如，如果执行指定 **ou=people, dc=example,dc=com** 的 LDAP 搜索请求，则搜索操作仅检查 dc=example,dc=com 目录树中的 **ou=people** 子树。

但是，不是所有条目都会自动返回以响应 LDAP 搜索，因为管理条目（具有 **ldapsubentry** 对象类）默认不会通过 LDAP 搜索返回。例如，管理对象是用来定义角色或服务类的条目。要在搜索响应中包含这些条目，客户端需要专门搜索具有 **ldapsubentry** 对象类的条目。有关服务类的更多信息，请参阅 第 4.3.2 节 "关于角色" 以了解有关角色和 第 5.3 节 "关于服务类" 的更多信息。

1.3.2. 分发目录数据

当目录树的不同部分存储在单独的数据库中时，目录可以并行处理客户端请求，从而提高性能。数据库甚至可以位于不同的机器上，以进一步提高性能。

分布式数据由目录的子树中的一个特殊条目（称为 *database link*）连接，它指向远程存储的数据。当客户端应用程序从数据库链接请求数据时，数据库链接会从远程数据库检索数据并将其返回到客户端。在此条目下尝试的所有 LDAP 操作都将发送到远程机器。此方法称为 *链*。

链 (Chaining) 作为插件在服务器中实施，默认被启用。

1.4. 目录设计概述

在实际部署前，规划目录服务是确保成功的关键。设计流程涉及收集有关目录要求的数据，如环境和数据源、用户和使用该目录的应用程序。此信息是设计有效目录服务的必要性，因为它有助于识别所需的安排和功能。

目录服务器的灵活性意味着可以重新处理目录设计以满足意外或更改的要求，即使在部署 Directory Server 后也是如此。

1.4.1. 设计过程概述

1. 第 2 章 *规划目录数据*

   目录包含诸如用户名、电话号码和组详情等数据。本章分析机构中各种数据源，并了解它们的相互关系。它描述了可以在目录中存储的数据类型和其他要执行的任务，以设计目录服务器的内容。

2. 第 3 章 *设计目录架构*

   目录旨在支持一个或多个启用了目录的应用程序。这些应用对存储在 目录中的数据的要求，如文件格式。目录架构决定了 目录中存储的数据的特征。目录服务器附带的标准架构在本章中引入，并介绍了如何定制架构的模式和提示，以维护一致性模式。

3. 第 4 章 *设计目录树*

   除了确定目录服务器中要包含*哪些信息*外，还需要确定*如何*组织和使用这些信息。本章介绍了目录树，并概述数据层次结构的设计。也提供了示例目录树设计。

4. 第 6 章 *设计目录拓扑*

   拓扑设计意味着目录树如何划分到多个物理目录服务器以及这些服务器如何与另一个服务器进行通信。设计的一般原则是，使用多个数据库，并使用一个有效的机制将分布式数据连接在一起。目录本身如何跟踪分布式数据的信息将在本章中阐述。

5. 第 7 章 *设计复制过程*

   使用复制时，多个目录服务器维护相同的目录数据，以提高性能并提供容错功能。本章论述了复制如何工作，即可以复制哪些数据类型、通用复制方案以及构建高可用性目录服务的提示。

6. 第 8 章 *设计同步*

红帽目录服务器中存储的信息可以通过与 Microsoft Active Directory 数据库中存储的信息同步，从而更好地与混合平台基础架构集成。本章论述了同步工作方式、什么数据可以同步，数据类型以及在目录数中的位置以适合于同步。

7. 第 9 章 *设计安全目录*

最后，计划如何保护目录中的数据并设计服务的其他方面，以满足用户和应用程序的安全要求。本章论述了常见的安全威胁、安全方法概述、分析安全需求过程中涉及的步骤，以及设计访问控制和保护目录数据完整性的建议。

1.4.2. 部署目录

部署目录服务器的第一步是安装测试服务器实例，以确保服务可以处理用户负载。如果服务在初始配置中不够，请调整设计并再次进行测试。调整设计，直到它成为一个稳定的服务，您可以放心地部署到您的实际企业环境中。

有关创建并实施目录试验的完整概述，请参阅 *了解和部署 LDAP 目录服务* （T. Howes、M. Smith、G. millan Technical Publishing, 1999）。

在创建并成功测试了目录服务器实例后，制订一个将目录服务迁移到生产环境中的计划，其需要涵盖以下注意事项：

- 所需资源的估算
- 计划需要完成的内容，以及时间
- 一组用于衡量部署是否成功的条件

有关管理和维护目录的信息，请参阅 *红帽目录服务器安装指南*，以了解安装目录服务和红帽目录服务器 *管理指南*。

1.5. 其他常规目录资源

以下发布提供有关目录、LDAP 和 LDIF 的详细有用信息：

- RFC 2849：LDAP 数据交换格式(LDIF)技术规格， http://www.ietf.org/rfc/rfc2849.txt
- RFC 2251：轻量级目录访问协议(v3)， http://www.ietf.org/rfc/rfc2251.txt
- *了解和部署 LDAP 目录服务*.T. Howes, M. Smith, G. am, Macmillan Technical Publishing, 1999.

所有红帽目录服务器文档（位于 https://access.redhat.com/documentation/zh-cn/red_hat_directory_server ）也包含有关使用 LDAP 和管理目录服务的高级别概念，以及特定于目录服务器的信息。

第 2 章 规划目录数据

保存在目录中的数据可能包括用户名、电子邮件地址、电话号码以及组用户的信息，或者可以包含其他类型的信息。目录中的数据类型决定了目录的结构、授予其访问数据的人员以及请求和授予此访问权限的方式。

本章论述了规划目录数据背后的问题和策略。

2.1. 目录数据简介

某些类型的数据更适用于目录服务。目录的理想数据具有以下特征：

- 它比写入更频繁。
- 它以 attribute-data 格式表示（例如，**surname=jensen**）。
- 这涉及到多个人或组。例如，许多人和应用程序都需要员工的名称或打印机的物理位置的信息。
- 它将从多个物理位置访问。

例如，一个员工对一个软件应用程序设置的首选选项可能并不适合于目录服务，因为只有单个应用程序实例需要访问这些信息。但是，如果应用程序能够从目录中读取首选项，并且用户可能希望根据不同站点的首选项与应用程序交互，那么在目录中包含首选项信息会非常有用。

2.1.1. 目录中包含的信息

任何关于个人或资产的描述性或有用的信息都可以作为属性添加到条目中。例如：

- 联系信息，如电话号码、物理地址和电子邮件地址。
- 描述性信息，如员工号码、工作标题、经理或管理员识别以及与作业相关的兴趣。
- 组织联系信息，如电话号码、物理地址、管理员标识和业务描述。
- 设备信息，如打印机物理位置、打印机类型以及打印机每分钟可以打印的页数。
- 有关公司交易合作伙伴、客户以及客户的联系和账单信息。
- 合同信息，如客户姓名、到期日期、工作说明和定价信息。
- 个人的软件首选项或软件配置信息。
- 资源站点，如指向 Web 服务器或特定文件或应用程序的文件系统。

将目录服务器用于服务器管理，需要规划其他类型的信息才能存储在 目录中。例如：

- 合同或客户端帐户详情
- 工资数据
- 物理设备信息
- 主页联系信息
- 企业内不同站点的办公室联系信息

2.1.2. 从目录中排除的信息

红帽目录服务器非常适合管理客户端应用程序读取和写入的大量数据，但不旨在处理大量非结构化对象，如镜像或其他媒体。这些对象应在文件系统中维护。但是，该目录可以使用指向 FTP、HTTP 和其他站点的指针 URL 将指针存储到这些种类的应用程序。

2.2. 定义目录需求

在设计目录数据时，不仅认为当前所需的数据，还认为目录（和组织）将随时间推移而变化。在设计过程中考虑目录的将来需求会影响目录中数据的构建和分发方式。

查看这些点：

- 现在应该把什么放置到目录中？
- 部署目录可解决哪些即时问题？
- 使用启用目录的应用程序的即时需求是什么？
- 在不久的将来，哪些信息会添加到目录中？例如，企业可能会使用目前不支持 LDAP 且在几个月内启用 LDAP 的核算软件包。识别 LDAP 兼容应用程序使用的数据，并在可行时计划将数据迁移到目录中。
- 将来可以将哪些信息保存在目录中？例如，托管公司可能使未来的客户与当前客户不同的数据要求，比如需要存储镜像或介质文件。虽然对未来的预测比较困难，但这可能会为您带来意外的好处。至少，这种计划有助于识别尚未考虑的数据源。

2.3. 执行站点问卷调查

站点调查是发现并标记目录中内容的正式方法。执行站点调查的时间预算低下，因为准备是目录架构的关键。站点调查由若干任务组成：

- 识别使用目录的应用程序。

  确定跨企业部署的目录应用程序及其数据需求。

- 识别数据源。

  调查企业并识别数据来源，如 Active Directory、其他 LDAP 服务器、PBX 系统、人工资源数据库和电子邮件系统。

- 对目录需要包含的数据进行定性。

  确定目录中应包括哪些对象（例如，人员或组）以及这些对象在目录中要维护哪些属性（如用户名和密码）。

- 确定要提供的服务级别。

  决定目录数据可用如何成为客户端应用程序，并相应地设计架构。要如何复制数据，以及将链策略配置为连接存储在远程服务器上的数据，从而如何影响复制数据。

  有关复制的信息，请参阅 第 7 章 *设计复制过程*，有关链的信息，请参阅 第 6.1 节 “拓扑概述”。

- 识别数据供应商.

数据供应商包含目录数据的主源。此数据可能会镜像到其他服务器，以进行负载平衡和恢复。对于每个数据，确定其数据的来源。

- 确定数据所有权。

  对于每个数据，确定负责确保数据最新状态的人员。

- 确定数据访问。

  如果数据从其他来源导入，为批量导入和增量更新制订相关策略。作为此策略的一部分，尝试在单个位置管理数据，并且限制可以更改数据的应用程序数量。另外，限制写入任何给定数据的人员数量。较小的组可确保数据完整性，同时减少管理开销。

- 记录网站调查。

由于目录可能会影响到多个机构，组成一个目录部署团队可能会是一个好的选择。这个团队可以包括来自受影响团队的人员，以执行站点调查。

公司通常都会拥有人工资源部门、会计部门或客户收帐部门、制造企业、销售组织和开发组织。包括这些机构中的代表可帮助调查流程。此外，直接涉及所有受影响的机构可以帮助构建接受从本地数据存储迁移到中央目录的接受操作。

2.3.1. 确定使用目录的应用程序

通常，访问目录的应用程序以及这些应用程序的数据需求会驱动对目录内容的规划。许多常见应用程序都使用目录：

- *目录浏览器应用程序，如在线笔记本电脑。*确定用户需要哪些信息（如电子邮件地址、电话号码和员工名称），并将其包含在目录中。

- *电子邮件应用程序，特别是电子邮件服务器.*所有电子邮件服务器都需要在目录中提供电子邮件地址、用户名和一些路由信息。其他系统可能还需要更高级的信息，如存储用户的邮箱数据、假期通知信息和协议信息（例如，IMAP 和 POP）等在磁盘中的位置。

- *启用目录的人工资源应用程序.*这需要更多个人信息，如政府身份标识号、家地址、家电话号码、出生日期、短文和职务。

- *Microsoft Active Directory*.通过 Windows User Sync，Windows 目录服务可以集成以与 Directory Server 配合使用。这两个目录都可以存储用户信息（用户名和密码、电子邮件地址、电话号码）和组信息（成员）。在现有 Windows 服务器部署后（反之亦然），使目录服务器数据可以平稳同步。

在检查要使用 目录的应用程序时，请查看每个应用所使用的信息类型。下表提供了应用程序示例以及每个应用程序所使用的信息：

表 2.1. 应用程序数据需要示例

Application（应用程序）	数据类别	data
电话	人员	名称、电子邮件地址、电话号码、用户 ID、密码、部门号码、经理、邮件停止.
Web 服务器	人员、组	用户 ID、密码、组名称、组成员、组所有者。

Application（应用程序）	数据类别	data
日历服务器	人员、会议室	名称、用户 ID、已用数字、会议室名称。

识别每个应用程序使用的应用程序和信息后，显然有些类型的数据会被多个应用程序使用。在数据规划阶段执行这种练习有助于避免目录中的数据冗余问题，并更清楚地显示数据与目录相关的应用程序需要的内容。

关于目录中维护的数据类型以及迁移到目录中的信息会受到这些因素影响的最终决定：

- 各种传统应用程序和用户所需的数据
- 传统应用程序与 LDAP 目录通信的功能

2.3.2. 识别数据源

要识别需要包括在目录中的所有数据，请执行对现有数据存储的调查。该调查应包括以下内容：

- 识别提供信息的机构。

  找到管理企业信息的所有组织。通常，这包括信息服务、人类资源、注册和会计部门。

- 识别作为信息源的工具和流程。

  某些信息的常见来源包括网络操作系统（Windows、Novell Netware、UNIX NIS）、电子邮件系统、安全系统、PBX（交换）系统和人类资源应用程序。

- 确定对数据进行中央化如何影响数据管理。

  集中式数据管理可能需要新的工具和新流程。对于一些机构，中央化可能需要增加员工，而对于其他一些机构，则可能会减少员工。

在调查期间，请考虑开发一个列表来标识企业中的所有信息源，类似于 表 2.2 “ 信息源示例” ：

表 2.2. 信息源示例

数据源	数据类别	data
人员资源数据库	人员	名称、地址、电话号码、部门号码、经理。
电子邮件系统	人员、组	名称、电子邮件地址、用户 ID、密码、电子邮件首选项.
设施系统	设施	构建名称、指纹、现有数字、访问代码。

2.3.3. 为目录数据定性

目录中标识的所有数据都可以根据以下通用点进行特征：

- 格式
- 大小

- 各种应用程序中发生次数
- 数据所有者
- 与其它目录数据的关系

研究目录中每个要包含的数据，以确定它与另一部分数据共享哪些特征。这有助于在 schema 设计阶段节省时间，详情请参阅 第 3 章 *设计目录架构*。

最好使用表，类似于 表 2.3 “目录数据特征”，它特征是目录数据。

表 2.3. 目录数据特征

data	格式	大小	所有者	相关
员工名称	文本字符串	128 个字符	人员资源	用户条目
传真号	电话号码	14 个数字	设施	用户条目
电子邮件地址	文本	多个字符	IS 部门	用户条目

2.3.4. 确定的服务质量等级

提供的服务级别取决于依赖支持目录的应用程序的用户的预期。要确定每个应用程序所需的服务级别，首先确定如何使用应用程序。

随着目录的演进，可能需要支持从生产到关键任务的各种服务级别。部署 目录后，可能会难以提高服务质量，因此请确保初始设计能够满足未来需求。

例如，如果必须消除总故障风险，请使用多层次配置，同一数据存在多个供应商。

2.3.5. 考虑数据供应商

数据供应商 是供应商数据源的服务器。任何相同的信息都存储在多个位置，都可能会降级数据完整性。数据供应商确保存储在多个位置的所有信息都是一致的且准确。有几个需要数据供应商的场景：

- 在目录服务器间复制
- Directory 服务器和 Active Directory 之间的同步
- 用于访问目录服务器数据的独立客户端应用程序

如果有应用程序与目录间接通信，请考虑数据的供应商源。保留更改数据的流程，以及可以更改数据的位置，尽可能简单。在决定单一站点管理数据后，使用相同的网站来管理该数据的所有数据。如果数据库在企业之间丢失同步，则单个站点简化了故障排除。

实现数据提供的方法有多种：

- 管理目录以及不使用目录的所有应用程序中的数据。

  维护多个数据供应商不需要自定义脚本在目录和其他应用程序中移动数据。但是，如果数据在一个位置更改，则人员必须在所有其他站点上进行更改。在目录中维护供应商数据以及所有不使用目录的应用程序都可能会导致数据在整个企业中无同步（这是应该阻止的目录是什么）。

- 在目录以外的一些应用程序中管理数据，然后编写脚本、程序或网关来将该数据导入到目录中。

  在非目录应用程序中管理数据时，如果两个应用程序已经用于管理数据，并且目录将仅用于查找（例如，在线企业电话书）。

数据的主副本是如何被维护的，这取决于特定目录的需求。但是，无论数据供应商如何维护，都保持简单且一致。例如，不要尝试管理多个站点中的数据，然后在竞争应用程序之间自动交换数据。这样做会导致"上次更改优先"情况，并增加了管理开销。

例如，该目录将管理员工的主页电话号码。LDAP 目录和人工资源数据库都存储此信息。人工资源应用程序是启用了 LDAP 的，因此应用程序可以编写它，自动将数据从 LDAP 目录传输到人工资源数据库，反之亦然。

试图管理该员工在 LDAP 目录和人力资源数据中的修改，但这意味着，在其他数据库中，电话号更改的最后一处将会覆盖信息。只要写入数据的最后应用程序都有正确的信息，这只可以接受。

如果该信息过期，或许因为从备份重新加载了人工资源数据，那么将删除 LDAP 目录中的正确电话号码。

通过多supplier 复制，目录服务器可以包含供应商信息源有关多个服务器的信息。多个供应商保持 changelogs，可以更加安全地解决冲突。有限数量的目录服务器被视为供应商，它们可以接受更改；然后，它们会将数据复制到复制服务器或消费者服务器。[1] 如果服务器脱机，则有超过数据供应商服务器提供安全故障转移。有关复制和多层复制的更多信息，请参阅 第 7 章 *设计复制过程*。

同步允许 Directory 服务器用户、组、属性和密码与 Microsoft Active Directory 用户、组、属性和密码集成。使用两个目录服务，决定他们是否处理相同的信息，这些信息量将共享，哪些服务将是该信息的数据供应商。最好的课程是选择单一应用程序来管理数据，并允许同步进程在其他服务上添加、更新或删除条目。

2.3.6. 确定数据所有权

*数据所有权*指的是负责确保数据最新状态的人员或组织。在数据设计阶段，决定谁可以向目录写入数据。以下是决定数据所有权的一些常见策略：

- 允许任何人对该目录的只读访问，但一组小的目录内容管理器除外。
- 允许个人用户自己管理某些战略性信息子集。

  此信息子集可能包括他们的密码、描述自身及其在机构内的角色信息、其自主权许可证的数量以及联系信息，如电话号码或办公室号码。

- 允许个人经理写入该人员信息的一些战略性子集，如联系信息或职位。
- 允许机构管理员创建和管理该组织的条目。

  这种方法允许组织管理员作为目录内容管理器运行。

- 创建赋予用户读取或写入访问权限组的角色。

  例如，可以为人工资源、财务或核算创建角色。允许这些角色具有读取访问权限、写入访问权限或这两个角色对组需要的数据具有读访问权限。这可包括工资信息、政府标识号以及主页电话号码和地址。

  有关角色和分组条目的更多信息，请参阅 第 4.3 节 “分组目录条目”。

可能有多个需要对相同信息进行写入访问权限的个人。例如，信息系统(IS)或目录管理组可能需要对员工密码进行写入访问权限。这可能要求员工本身具有他们自己的密码的写权限。虽然通常，多个人对同一信息具有写入权限，但尽量使该组保持小且易于识别。使组 small 有助于确保数据完整性。

有关为目录设置访问控制的详情请参考 第 9 章 *设计安全目录*。

2.3.7. 确定数据访问

确定数据所有权后，决定谁可以读取每个数据。例如，员工的主页电话号码可以存储在 目录中。这些数据对于许多组织（包括员工的经理和人力资源）非常有用。员工应该能够读取该信息以进行验证。但是，家联系信息可被视为敏感性，因此企业可能不能广泛使用。

对于目录中存储的每个信息，决定以下内容：

- 数据是否匿名读取？

  LDAP 协议支持匿名访问，并允许轻松查找常见信息，如办公室站点、电子邮件地址和业务电话号码。但是，匿名访问可让任何人访问目录访问通用信息。因此，请小心使用匿名访问。

- 数据是否可以跨企业广泛读取？

  可以设置访问控制，使客户端必须登录（或绑定到）目录才能读取特定信息。与匿名访问不同，这种形式的访问控制可确保只有机构的成员可以查看目录信息。它还捕获目录访问日志中的登录信息，因此有记录谁访问信息。

  有关访问控制的更多信息，请参阅 第 9.7 节 “设计访问控制”。

- 是否存在一组需要读取数据的人员或应用程序的可识别组？

  对数据具有写入权限的用户通常需要读访问权限（具有密码的写入访问权限除外）。还可能有特定于特定组织或项目组的数据。识别这些访问权限需要有助于确定哪些组、角色和访问控制。

  有关组和主机的详情，请参考 第 4 章 *设计目录树*。有关访问控制的详情请参考 第 9.7 节 “设计访问控制”。

对每个目录数据进行这些决策会定义目录的安全策略。这些决策取决于站点的性质以及已在站点所提供的各种安全类型。例如，允许防火墙或无法直接访问互联网意味着，如果目录直接放置在互联网上，就很难支持匿名访问。此外，某些信息可能只需要访问控制和验证措施来限制访问被正确；可能需要在数据库中加密其他敏感信息，因为它存储在数据库中。

在很多国家/地区，数据保护法律规定企业必须如何维护个人信息，并限制谁有权访问个人信息。例如，法律可能会禁止匿名地址和电话号码访问，或者可能要求用户能够查看和纠正代表它们的条目中的信息。请务必与机构的法律部门进行检查，以确保目录部署遵循企业运营的国家/地区的必要法律。

在 第 9 章 *设计安全目录* 中详细介绍创建安全策略及其实现方式。

2.4. 记录网站调查

由于数据设计的复杂性，记录站点调查的结果。站点调查的每个步骤都可使用简单的表来跟踪数据。考虑构建概述决策和卓越的关注的供应商表。好的提示是使用电子表格，以便可以轻松地对表的内容进行排序和搜索。

表 2.4 “示例：标记数据所有权和访问” 标识站点调查所标识每个数据的所有权和数据访问权限。

表 2.4. 示例：标记数据所有权和访问

数据名称	所有者	供应商服务器/应用程序	自助读取/写	全局读	HR Writable	IS Writable
员工名称	HR	PeopleSoft	只读	是（匿名）	是	是
用户密码	IS	Directory US-1	读/写	否	否	是
主页电话号码	HR	PeopleSoft	读/写	否	是	否
员工位置	IS	Directory US-1	只读	是（必须登录）	否	是
办公室电话号码	设施	电话交换机	只读	是（匿名）	否	否

表中的每一行显示正在评估哪些信息类型、哪些部门对其兴趣以及如何使用和访问信息。例如，在第一行中，*employee names* 数据具有以下管理注意事项：

- *Owner*。人类资源拥有此信息，因此负责更新和更改。
- *Supplier Server/Application*。PeopleSoft 应用程序管理员工名称信息。
- *Self Read/Write*。个人可以读取自己的名称，但不能对它进行写（或更改）。
- *Global Read*。员工名称可以被有权访问该目录的任何人匿名读取。
- *HR Writable*.人员资源组成员可能会更改、添加和删除目录中的员工名称。
- *IS 可写*.information services 组的成员可在 目录中更改、添加和删除员工名称。

2.5. 重复站点问卷调查

可能需要多次进行站点调查，特别是当一个企业在多个城市或国家设有办事处时。信息性需求可能很复杂，有些不同组织必须将其信息保存在本地办事处，而不是单一的集中站点。

在这种情况下，每个办公室都会保留一份主要信息副本才能执行自己的站点调查。在完成站点调查过程后，每个调查的结果都应该返回至中央团队（包括每个办公室的代表），以便在整个企业的数据架构模型和目录树的设计中使用。

[1] 在复制中，*使用者 服务器或副本* 服务器是一个从供应商服务器或 hub 服务器接收更新的服务器。

第 3 章 设计目录架构

在 第 2 章 *规划目录数据* 中进行的站点问卷调查显示保存在目录中数据的信息。目录 *架构* 描述了目录中的数据类型，因此决定使用什么 schema 反映如何表示目录中存储的数据。在架构设计过程中，每个数据元素映射到 LDAP 属性，相关的元素将收集到 LDAP 对象类中。设计良好的架构有助于保持目录数据的完整性。

本章论述了目录架构，以及如何针对独特的组织需求设计模式。

有关复制 schema 的详情，请参考 第 7.4.4 节 “模式复制”。

3.1. 模式设计过程概述

在架构设计过程中，选择并定义用于表示由 Red Hat Directory Server 存储的条目的对象类和属性。模式设计涉及以下步骤：

1. 选择预定义的架构元素以满足尽可能多的数据需求。
2. 扩展标准目录服务器架构，以定义新的元素以满足其他剩余需求。
3. 规划架构维护。

最简单且易于维护的选项是使用由 Directory Server 提供的标准 schema 中定义的现有 schema 元素。选择标准架构元素有助于确保与启用了目录的应用程序兼容。由于该架构基于 LDAP 标准，因此已审核并同意很多目录用户。

3.2. 标准架构

目录架构通过对数据值的大小、范围和格式的限制来维护目录中存储的数据的完整性。该架构反映了目录包含哪些类型（如人员、设备和机构）以及每个条目可用的属性的决策。

Directory Server 中包含的预定义模式包含标准的 LDAP 模式，以及其他特定于应用程序的模式，以支持服务器的功能。虽然此架构满足大多数目录需求，但可将新的对象类和属性添加到架构（*extending* schema）中以满足目录的唯一需求。有关扩展 schema 的信息，请参阅 第 3.4 节 “自定义架构”。

3.2.1. 模式格式

目录服务器将其模式格式基于 LDAP 协议的版本 3。此协议要求目录服务器通过 LDAP 本身发布其模式，允许目录客户端应用以编程方式检索架构并相应地调整其行为。Directory 服务器的全局模式集合可在 **cn=schema** 条目中找到。

目录服务器模式与 LDAPv3 模式稍有不同，因为它使用自己的专有对象类和属性。另外，它使用 schema 条目中的私有字段，名为 **X-ORIGIN**，它描述了最初定义 schema 条目的位置。

例如，如果在标准 LDAPv3 模式中定义了模式条目，则 **X-ORIGIN** 字段指的是 RFC 2252。如果红帽为目录服务器的使用定义了该条目，则 **X-ORIGIN** 字段包含值 **Netscape Directory Server**。

例如，标准 **person** 对象类出现在架构中，如下所示：

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

此架构条目指出对象标识符或 *OID*，类(**2.5.6.6**)、对象类的名称(**个人**)、类的描述(**Standard Person**)，然后列出所需的属性(***objectclass***,***sn***, 和 ***cn***)，以及允许的属性(***描述***、***seeAlso***、***telephoneNumber*** 和 ***userPassword***)。

有关 LDAPv3 模式格式的更多信息，请参阅 LDAPv3 属性语法定义文档、RFC 2252 和其他标准架构定义，如 RFC 247、RFC 2927 和 RFC 2307。Red Hat Directory Server 支持所有这些架构元素。

3.2.2. 标准属性

属性包含特定的数据元素，如名称或传真号码。目录服务器将数据表示为*属性数据对*，它是一个与特定信息片段关联的描述性 schema 属性。它们也称为 *attribute-value assertions* 或 AVAs。

例如，目录可以将一个数据（如个人的名称）存储在带有标准属性的对中，本例中为 ***commonName***(***cn***)。因此，名为 Babs Jensen 的人员的条目具有 attribute-data 对 **cn: Babs Jensen**。

实际上，整个条目都以一系列属性对的形式表示。Babs Jensen 的整个条目如下：

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

Babs Jensen 的条目包含某些属性的多个值。***givenName*** 属性显示两次，每次都有一个唯一值。

在 schema 中，每个属性定义包含以下信息：

- 唯一的名称。
- 属性的对象标识符(OID)。
- 属性的文本描述。
- 属性语法的 OID。
- 指示 属性是单值还是多值，无论属性是否适用于目录自己的用途、属性的来源，以及与 属性关联的任何其他匹配规则。

例如，***cn*** 属性定义出现在 schema 中，如下所示：

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

属性的语法定义属性允许的值的格式。在某种程度上，语法有助于定义可以存储在 属性中的信息类型。Directory 服务器支持所有标准属性语法。

受支持的 LDAP 属性语法包括在 红帽目录服务器 10 配置、命令和文件参考的 *Directory Server* Attribute Syntaxes 中。

3.2.3. 标准对象类

对象类用于对相关信息进行分组。通常，对象类代表一个实际对象，如个人或传真机器。在可以使用对象类及其目录中的属性之前，必须在架构中识别它。目录默认识别对象类的标准列表；它们列在 *Red Hat Directory Server Configuration、命令和文件参考* 中。

每个目录条目都至少属于一个对象类。将一个对象类放在条目上的 schema 中可告知 Directory 服务器，该条目可以具有一组特定的可能属性值，并且必须具有另一个、通常更小的、必需属性值集。

对象类定义包含以下信息：

- 唯一的名称。
- 为 *对象命名的对象标识符* (OID)。
- 一组强制属性。
- 组允许（或可选）属性。

例如，标准 **person** 对象类出现在架构中，如下所示：

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

对于所有 Directory Server 的 schema，对象类会被定义并直接存储在 Directory Server 中。这意味着，使用标准 LDAP 操作可以查询和更改目录的模式。

3.3. 将数据映射到默认架构

在站点问卷调查中识别的数据（如 第 2.3 节 “执行站点问卷调查” 所述）必须映射到现有的默认目录模式。这部分论述了如何查看现有的默认模式，并提供了一种将数据映射到适当的现有 schema 元素的方法。

如果架构中有一个与现有默认架构不匹配的元素，请创建自定义对象类和属性。如需更多信息，请参阅 第 3.4 节 “自定义架构”。

3.3.1. 查看默认目录架构

默认目录模式存储在 **/usr/share/dirsrv/schema/** 目录中。

此目录包含 Directory 服务器的所有通用模式。LDAPv3 标准用户和组织模式可在 **00core.ldif** 文件中找到。早期版本的目录所使用的配置模式可在 **50ns-directory.ldif** 文件中找到。

警告

不要修改 默认目录模式。

有关 目录中找到的每个对象类和属性的更多信息，请参阅 *Red Hat Directory Server Configuration、命令和文件参考*。相同的指南还提供有关架构文件和目录配置属性的更多信息。

3.3.2. 将数据与架构元素匹配

现在，站点问卷调查中标识的数据需要映射到现有的目录 schema。这个过程涉及以下步骤：

1. 确定数据描述的对象类型。

   选择一个最能与站点调查中描述的数据匹配的对象。有时，一个数据可以描述多个对象。确定在目录 schema 中是否需要记录区别。

   例如，电话号码可以描述员工的电话号码和会议室的电话号码。确定这些不同类型的数据是否需要被视为目录架构中不同的对象。

2. 从默认架构中选择一个类似的对象类。

   最好使用通用对象类，如组、人员和机构。

3. 从匹配的对象类选择类似的属性。

   从匹配对象类中选择一个属性，它们最适合站点调查中标识的数据片段。

4. 识别来自站点调查的不匹配数据。

   如果有一些数据与默认目录架构定义的对象类和属性不匹配，请自定义 schema。如需更多信息，请参阅 第 3.4 节 “自定义架构”。

例如，下表将目录模式元素映射到在站点问卷调查 第 2 章 *规划目录数据* 中标识的数据：

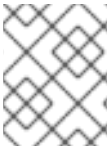
表 3.1. 数据映射到默认目录 Schema

data	所有者	对象类	属性
员工名称	HR	个人	cn (commonName)
用户密码	IS	个人	userPassword
主页电话号码	HR	inetOrgPerson	homePhone
员工位置	IS	inetOrgPerson	localityName
办公室电话号码	设施	个人	telephoneNumber

在 表 3.1 “数据映射到默认目录 Schema” 中，员工名称描述一个人。在默认目录架构中，有一个 **person** 对象类，它继承自 **顶级** 对象类。此对象类允许多个属性，其中一个是 ***cn*** 或 ***commonName*** 属性来描述个人的完整名称。此属性为包含员工名称数据的最佳匹配。

用户密码还描述了 **person** 对象类的一个方面，而 ***userPassword*** 属性则列在 **person** 对象类的允许属性中。

主页电话号码描述了个人的一个方面，但是，与 **person** 对象类关联的列表中没有相关的属性。主页电话号码描述了企业网络中个人的方面。此对象与目录 schema 中的 **inetOrgPerson** 对象类对应。**inetOrgPerson** 对象类继承自 **organizationPerson** 对象类，后者又从 **person** 对象类继承。**inetOrgPerson** 对象允许的属性是 ***homePhone*** 属性，它适用于包含员工的主页电话号码。



注意

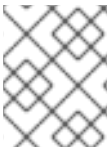
Red Hat Directory Server 配置、命令和文件参考 可用于决定哪些属性可用于您的数据。每个属性都列出接受它的对象类，每个对象类被列为必需属性和允许的属性。

3.4. 自定义架构

如果标准模式对于目录需求太小，则可以进行扩展。Directory 服务器中的 Web 控制台可用于通过轻松添加属性和对象类来扩展 schema。也可以创建 LDIF 文件并手动添加 schema 元素。如需更多信息，请参阅 *红帽目录服务器管理指南*。

在自定义 Directory 服务器模式时请注意以下规则：

- 使架构尽可能简单。
- 尽可能重复使用现有的 schema 元素。
- 尽可能减少为每个对象类定义的必要属性数量。
- 不要为相同目的（数据）定义多个对象类或属性。
- 不要修改属性或对象类的任何现有定义。



注意

在自定义 schema 时，*从* 删除或替换标准模式。这样做可能会导致与其他目录或其他 LDAP 客户端应用程序兼容性。

自定义对象类和属性在 **99user.ldif** 文件中定义。每个实例在 **/etc/dirsrv/slapd-*instance_name*/schema/** 目录中维护自己的 **99user.ldif** 文件。也可以创建自定义模式文件，并将架构动态重新加载到服务器中。

3.4.1. 当扩展架构时

虽然 Directory 服务器提供的对象类和属性应该满足最常见的公司需求，但给定对象类可能无法存储有关机构的专门信息。此外，该架构可能需要扩展来支持启用了 LDAP 的应用的唯一数据需要的对象类和属性。

3.4.2. 获取和分配对象标识符

必须为每个 LDAP 对象类或属性分配一个唯一名称和 *对象标识符* (OID)。当定义了 schema 时，元素需要一个基础 OID，它与您的机构是唯一的。个 OID 足以满足所有架构需求。只需添加另外一种层次结构级别，为属性和对象类创建新的分支。在 schema 中获取和分配 OID 涉及以下步骤：

1. 从互联网编号分配机构(IANA)或国家组织获取 OID。

   在某些国家/地区，公司已分配给他们的 OID。如果您的组织还没有 OID，可以从 IANA 获取。有关更多信息，请访问 IANA 网站 http://www.iana.org/cgi-bin/enterprise.pl。

2. 创建 OID registry 以跟踪 OID 分配。

   OID registry 是目录 schema 中使用的 OID 和描述的列表。这样可确保不使用 OID 用于多个目的。然后使用 schema 发布 OID 注册表。

3. 在 OID 树中创建分支以容纳 schema 元素。

在 OID 分支或目录 schema 下至少创建两个分支，将 *OID*.**1** 用于属性，*OID*.**2** 用于对象类。要定义自定义匹配规则或控制，请根据需要添加新分支（例如*OID*.**3**）。

3.4.3. 命名属性和对象类

为新属性和对象类创建名称时，请尽可能有意义的名称。这样，可以更轻松地将架构用于 Directory 服务器管理员。

通过在所有 schema 元素上包含唯一前缀，避免架构元素和现有 schema 元素间的命名冲突。例如，Example Corp. 可能会在每个自定义 schema 元素前添加前缀 **示例**。它们可能会添加一个名为 **examplePerson** 的特殊对象类，以识别其目录中的 Example Corp. employees。

3.4.4. 定义新对象类的策略

创建新对象类的方法有两种：

- 创建多个新对象类，每个对象类结构对应一个属性。
- 创建一个对象类，它支持为目录创建的所有自定义属性。这种对象类的方法是将它定义为辅助对象类。

这两种方法可能最容易。

例如，假设管理员希望创建属性 ***exampleDateOfBirth***、***examplePreferredOS***、***exampleBuildingFloor*** 和 ***exampleViceP 常处***。简单的解决方案是创建多个对象类，允许其中的一些部分属性。

- 一个对象类(**examplePerson**)被创建，并允许 ***exampleDateOfBirth*** 和 ***examplePreferredOS***。**examplePerson** 的父项是 **inetOrgPerson**。
- 第二个对象类 **exampleOrganization** 允许 ***exampleBuildingFloor*** 和 ***exampleViceP*** 常常。**exampleOrganization** 的父是 **机构** 对象类。

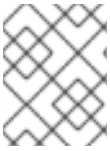
新对象类以 LDAPv3 模式格式显示，如下所示：

```
objectclasses: ( 2.16.840.1.117370.999.1.2.3 NAME 'examplePerson' DESC 'Example Person
Object Class'
    SUP inetorgPerson MAY (exampleDateOfBirth $ examplePreferredOS) )

objectclasses: ( 2.16.840.1.117370.999.1.2.4 NAME 'exampleOrganization' DESC 'Organization
Object Class'
    SUP organization MAY (exampleBuildingFloor $ exampleVicePresident) )
```

或者，创建一个允许所有这些属性的单一对象类，并将其与需要这些属性的任何条目一起使用。单个对象类如下所示：

```
objectclasses: (2.16.840.1.117370.999.1.2.5 NAME 'exampleEntry' DESC 'Standard Entry Object
Class' SUP top
    AUXILIARY MAY (exampleDateOfBirth $ examplePreferredOS $ exampleBuildingFloor $
exampleVicePresident) )
```

新的 **exampleEntry** 对象类被标记为 **AUXILIARY**，这意味着它可以与任何条目一起使用，而不考虑其 structural 对象类。



注意

示例中的新对象类的 OID (**2.16.840.1.117370**)基于以前的 Netscape OID 前缀。要创建自定义对象类，请获取 第 3.4.2 节 “获取和分配对象标识符” 所述的 OID。

根据组织环境，可以通过几种不同的方法来组织新对象类。在决定如何实施新对象类时请考虑以下几点：

- 多个对象类会导致更多架构元素来创建和维护。

  通常，元素数量较小且需要较少的维护。但是，如果架构中增加了两个或多个对象类，那么使用单个对象类可能更易于使用。

- 多个对象类需要更小心、更严格的数据设计。

  严格数据设计强制注意用于放置每个数据的对象类结构，这可能很有帮助或繁琐。

- 当数据被应用到多个对象类（如人员和资产条目）时，单一对象类简化了数据设计。

  例如，可以在 person 和 group 条目上设置自定义 ***preferredOS*** 属性。单个对象类可以在这两类条目上允许此属性。

- 避免新对象类的必要属性。

  指定 **require** 而不是 **允许** 新对象类中的属性，可能会导致 schema 不灵活。在创建新对象类时，请使用 **allow** 而不是 **需要** 尽可能多的内容。

  定义新对象类后，决定其允许和需要哪些属性，以及它继承属性的对象类。

3.4.5. 定义新属性的策略

对于应用程序兼容性和长期维护，尽量尝试使用标准属性。搜索默认目录架构中已存在的属性，并使用它们与新对象类关联或签出 *Directory Server Schema 指南*。但是，如果标准模式不包含您需要的所有信息，则添加新的属性和新对象类。

例如，个人条目可能需要超过 **人员**、**机构管理**员 或 **inetOrgPerson** 对象类支持的属性。例如，标准目录服务器架构中不存在任何属性来存储 birth 日期。可以创建新属性 ***dateOfBirth***，并在新的辅助对象类 **examplePerson** 中设置为允许的属性。

```
attributetypes: ( dateofbirth-oid NAME 'dateofbirth' DESC 'For employee birthdays'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'Example defined')

objectclasses: ( 2.16.840.1.117370.999.1.2.3 NAME 'examplePerson' DESC 'Example Person
Object Class'
    SUP inetorgPerson MAY (exampleDateOfBirth $ cn) X-ORIGIN 'Example defined')
```

要记住的一个重要事项： *从不* 添加或删除自定义属性到标准架构元素。如果目录需要自定义属性，请添加自定义对象类来包含它们。

3.4.6. 删除架构元素

不要删除 Directory Server 默认包含的架构元素。未使用的模式元素代表没有操作或管理开销。删除标准 LDAP 模式的部分可能会导致目录服务器和其他启用了目录的应用程序安装间的兼容性问题。

但是，未使用的自定义模式元素可以被删除。在从 schema 中删除对象类定义前，使用对象类修改每个条目。首先删除定义可能会阻止使用对象类的条目被修改。修改条目的 schema 检查也会失败，除非从条目中删除了未知对象类值。

3.4.7. 创建自定义架构文件

除了由 Directory Server 提供的 **99user.ldif** 文件外，管理员还可以为 Directory 服务器创建自定义架构文件。这些架构文件保存特定于组织的新自定义属性和对象类。新的模式文件应位于 schema 目录中，**/etc/dirsrv/slapd-*instance_name*/schema/**。

所有标准属性和对象类仅在加载自定义 schema 元素后加载。

注意

自定义架构文件不应以数字方式或字母顺序高于 **99user.ldif**，或者服务器可能会遇到问题。

在创建自定义模式文件后，可通过两种方式在所有服务器中分发架构更改：

- 手动将这些自定义模式文件复制到实例的 schema 目录 **/etc/dirsrv/slapd-*instance*/schema**。要载入架构，请通过运行 **schema-reload.pl** 脚本来动态重新加载模式。
- 使用 LDAP 客户端（如 Web 控制台或 **ldapmodify** ）修改服务器上的 schema。
- 如果服务器被复制，则允许复制过程将架构信息复制到每个使用者服务器。

  通过复制，所有复制模式元素都复制到消费者服务器的 **99user.ldif** 文件中。要将架构保留在自定义模式文件中，如 **90example_schema.ldif**，必须手动将文件复制到消费者服务器。复制不会复制架构文件。

如果这些自定义模式文件没有复制到所有服务器，则当供应商服务器上的使用 LDAP 客户端（如 Web 控制台或 **ldapmodify** ）更改时，模式信息才会复制到副本（消费者服务器）。

当架构定义复制到不存在的消费者服务器时，它们存储在 **99user.ldif** 文件中。目录无法跟踪存储架构定义的位置。在消费者的 **99user.ldif** 文件中存储架构元素不会产生问题，只要该架构仅在供应商服务器上维护。

如果自定义架构文件被复制到每台服务器上，必须再次复制到每台服务器上更改架构文件。如果没有再次复制文件，则更改可能会复制并存储在消费者的 **99user.ldif** 文件中。在 **99user.ldif** 文件中进行了更改可能会导致架构管理困难，因为一些属性将出现在消费者上的两个独立模式文件中，一次在从供应商复制的原始自定义模式文件中，并在复制后再次出现在 **99user.ldif** 文件中。

有关复制模式的更多信息，请参阅 第 7.4.4 节 “模式复制”。

3.4.8. 自定义架构最佳实践

使用架构文件时，请确保创建兼容且易于管理的 schema。

3.4.8.1. 命名架构文件

在命名自定义模式文件时，使用以下命名格式：

```
[00-99]yourName.ldif
```

将自定义架构文件命名为 lower （按字母和字母顺序）而不是 **99user.ldif**。这可让目录服务器通过 LDAP 工具和 Web 控制台写入 **99user.ldif**。

99user.ldif 文件包含 **X-ORIGIN** 值为 **'user defined'** 的属性；但是，目录服务器将所有"用户定义的"模式元素写入最高命名的文件，然后按字母顺序排列。如果存在一个名为 **99zzz.ldif** 的模式文件，下一次更新 schema 时（通过 LDAP 命令行工具或 Web 控制台）所有带有 **X-ORIGIN** 的值为 **'user defined'** 的属性都会写入 **99zzz.ldif**。结果是两个包含重复信息的 LDIF 文件，而 **99zz.ldif** 文件中的一些信息可能会被清除。

3.4.8.2. 使用"用户定义的"作为 Origin

不要在 自定义架构文件的 **X-ORIGIN** 字段中使用 **'user defined'** （如 **60example.ldif**），因为当通过 LDAP 添加模式时，目录服务器在内部使用 **'user defined'**。在自定义架构文件中，使用更多描述性，如 **'Example Corp. defined'**。

但是，如果自定义架构元素直接添加到 **99user.ldif** 中，请使用 **'user defined'** 作为 **X-ORIGIN** 的值。如果设置了不同的 **X-ORIGIN** 值，则服务器可能会覆盖它。

使用值 **'user defined'** 的 **X-ORIGIN** 可确保 **99user.ldif** 文件中的模式定义不会被 Directory 服务器从文件中删除。目录服务器不会删除它们，因为它依赖于值 **'user defined'** 的 **X-ORIGIN** 来告知它哪些元素应驻留在 **99user.ldif** 文件中。

例如：

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

在 Directory Server 加载 schema 条目后，它如下所示：

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

3.4.8.3. 在对象类前定义属性

在添加新的架构元素时，需要定义所有属性，然后才能在对象类中使用它们。属性和对象类可以在相同的架构文件中定义。

3.4.8.4. 在单个文件中定义架构

每个自定义属性或对象类都应该只在一个 schema 文件中定义。这可防止服务器在加载最早创建的模式时覆盖任何以前的定义（因为服务器首先载入数字顺序，然后按字母顺序载入该 schema）。决定如何在重复的文件中保留 schema：

- 请注意每个架构文件中包括哪些 schema 元素。
- 在命名和更新模式文件中要小心。通过 LDAP 工具编辑模式元素时，更改会自动写入最后一个文件（通常）。大多数架构更改，然后写入默认文件 **99user.ldif**，而不是自定义架构文件，如 **60example.ldif**。另外，**99user.ldif** 中的架构元素会覆盖其他架构文件中的重复元素。
- 将所有架构定义添加到 **99user.ldif** 文件。这在通过 Web 控制台管理模式时很有用。

3.5. 维护一致性架构

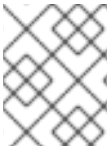
目录服务器中的一致性模式可帮助 LDAP 客户端应用程序查找目录条目。使用不一致的 schema 将很难在目录树中高效地查找信息。

不一致的 schema 使用不同的属性或格式来存储相同的信息。通过以下方法维护 schema 一致性：

- 使用 schema 检查来确保属性和对象类符合 schema 规则。
- 使用语法验证，以确保属性值与所需属性语法匹配。
- 选择并应用一致的数据格式。

3.5.1. Schema 检查

架构检查可确保所有新的或修改的目录条目均符合 schema 规则。当违反规则时，目录会拒绝请求的更改。



注意

架构检查仅检查是否存在正确的属性。要验证属性值是否正确语法，请使用语法验证，如第 3.5.2 节 “语法验证” 所述。

默认情况下，目录启用 schema 检查。红帽建议不要禁用此功能。有关启用和禁用模式检查的详情，请参考 *红帽目录服务器管理指南*。

启用架构检查后，对于对象类定义的必需属性不利。对象类定义通常至少包含一个必需属性以及一个或多个可选属性。可选属性是可以是的属性，但不需要添加到目录条目中。尝试向条目的对象类定义没有必要或允许的条目中添加属性会导致 Directory 服务器返回对象类违反消息。

例如，如果将一个条目定义为使用 **organizationalPerson** 对象类，则该条目需要通用名称(***cn***)和 surname (***sn***)属性。也就是说，创建条目时必须设置这些属性的值。另外，还有可在条目上使用的属性列表，包括 ***telephoneNumber***、***uid***、***streetAddress*** 和 ***userPassword*** 等描述性属性。

3.5.2. 语法验证

语法验证 意味着 Directory 服务器检查属性值是否与该属性的所需语法匹配。例如，语法验证将确认一个新的 ***telephoneNumber*** 属性实际具有其值的有效电话号码。

3.5.2.1. 语法验证概述

默认情况下启用语法验证。这是最基本的语法验证。与架构检查一样，这将验证任何目录修改并拒绝违反语法规则的更改。可以选择性地配置其他设置，使语法验证可以记录关于语法违规消息，然后拒绝修改或允许修改过程成功。

语法验证会检查添加新属性值的 LDAP 操作，因为添加了新的属性或因为属性值已更改。语法验证不会处理通过像复制一样通过数据库操作添加的现有属性或属性。可以使用特殊脚本 **syntax-validate.pl** 验证现有属性。

这个功能会验证所有属性语法，但二进制语法（无法验证）和非标准语法（没有定义所需的格式）和非标准语法除外。语法是根据 RFC 4514 （DN 之外）进行验证，该语法针对不太严格的 RFC 1779 或 RFC 2253 进行验证。（可以配置多个 DN 验证。）

3.5.2.2. 语法验证和其他目录服务器操作

语法验证对于标准 LDAP 操作主要相关，如创建条目（添加）或编辑属性（修改）。验证属性语法可能会影响其他目录服务器操作。

数据库加密

对于普通的 LDAP 操作，属性仅在值写入数据库之前加密。这意味着，加密会在验证属性语法*后*进行。

可以导出并导入加密的数据库（如 第 9.8 节 "加密数据库"所述）。通常，强烈建议使用 **db2ldif** 和 **ldif2db** 的 **-E** 标志完成这些导出和导入操作，这允许导入操作正常进行。但是，如果加密数据库在没有使用 **-E** 标志（不支持）的情况下导出，那么会创建一个带有加密值的 LDIF。当导入此 LDIF 后，无法验证加密的属性，则会记录警告，并在导入的条目中跳过属性验证。

同步

对于 Windows Active Directory 条目和 Red Hat Directory Server 条目中的属性，允许或拒绝语法存在区别。在这种情况下，Active Directory 值无法正确同步，因为语法验证会在 Directory Server 条目中强制实施 RFC 标准。

复制

如果 Directory Server 11.0 实例是一个将更改复制到消费者的供应商，则使用语法验证没有问题。但是，如果复制中的供应商是旧版本的目录服务器或已禁用语法验证，则 11.0 消费者上不应使用语法验证，因为目录服务器 11.0 使用者可能会拒绝供应商允许的属性值。

3.5.3. 选择一致性数据格式

LDAP 模式允许任何数据放置在任何属性值中。但是，务必要通过选择适合 LDAP 客户端应用程序和目录用户的格式，将数据存储在目录树中。

使用 LDAP 协议和目录服务器时，数据必须使用 RFC 2252 中指定的数据格式表示。例如，两个 ITU-T 建议中定义了正确的 LDAP 格式用于电话号码：

- *ITU-T 建议 E.123*.国家和国际电话号码表示法。
- *ITU-T 建议 E.163*.国际电话服务的数字计划。例如，美国电话号码格式化为 **+1 555 222 1717**。

再如，***postalAddress*** 属性需要一个以多行字符串形式的属性值，该形式使用美元符号($)作为行分隔符。正确格式化的目录条目如下：

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

属性可以要求字符串、二进制输入、整数和其他格式。允许的格式在属性的 schema 定义中设置。

3.5.4. 维护复制架构

在编辑目录架构时，更改会记录在 changelog 中。在复制过程中，对更改进行扫描后，会复制任何更改。在复制模式下保持一致性，复制可以平稳进行。考虑在复制环境中保持一致的模式：

- 不要修改只读副本中的 schema。

  在只读副本中修改 schema 在 schema 中引入了不一致的问题，并导致复制失败。

- 不要创建具有相同名称的属性，它们使用不同的语法。

  如果在读写副本中创建属性，其名称与供应商副本上的属性相同，但与供应商的属性不同，复制将失败。

3.6. 其他架构资源

有关标准 LDAPv3 模式的更多信息，请参见以下链接：

- RFC 2251：轻量级目录访问协议(v3)， http://www.ietf.org/rfc/rfc2251.txt
- RFC 2252: LDAPv3 属性语法定义, http://www.ietf.org/rfc/rfc2252.txt
- X.500 用户架构的 RFC 2256: Summary for with LDAPv3, http://www.ietf.org/rfc/rfc2256.txt
- Internet 工程任务组(IETF) http://www.ietf.org/
- *了解和部署 LDAP 目录服务*.T. Howes, M. Smith, G. am, Macmillan Technical Publishing, 1999.

第 4 章 设计目录树

*目录树*提供了一种查看由目录服务存储的数据的方法。存储在目录中的信息类型、企业的物理性质、用于目录的应用程序以及所实施的复制类型形成了目录树的设计。

本章概述了设计目录树的步骤。

4.1. DIRECTORY TREE 简介

目录树提供了一种方式，用于命名的目录数据，并由客户端应用程序引用。目录树与其他设计决策紧密交互，包括可用的分发、复制或控制对目录数据的访问。在部署之前，投资时间来正确地设计目录树。正确设计的目录树可以在部署阶段节省大量时间和精力，之后目录服务运行完毕。

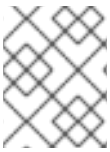
精心设计好的目录树提供以下内容：

- 简化的目录数据维护。
- 创建复制策略和访问控制的灵活性。
- 支持使用目录服务的应用程序。
- 目录用户的简化目录导航。

目录树的结构遵循分层 LDAP 模型。目录树提供了一种方式，以不同的逻辑方式组织数据，如组、人员或位置。它还决定如何在多个服务器间对数据进行分区。例如，每个数据库都需要在后缀级别上对数据进行分区。如果没有正确的目录树结构，它可能无法有效地将数据分散到多个服务器上。

此外，复制受所用的目录树类型的限制。仔细定义分区以使复制工作。要仅复制目录树的部分，请在设计过程中考虑这一点。

要使用分支点上的访问控制，也请考虑在目录树设计中。



注意

目录服务器支持分层导航和组织（称为虚拟目录信息树视图）的概念。在设计目录树前，请查看 第 4.4 节 “虚拟目录信息树视图”。

4.2. 设计目录树

在目录树设计中计划有几个主要决策：

- 选择包含数据的后缀。
- 确定数据条目之间的分层关系。
- 命名目录树层次结构中的条目。

4.2.1. 选择 Suffix

后缀是目录树根目录下的条目名称，并且目录数据存储在其中。目录可以包含多个后缀。如果有两个或者多个目录树没有自然常见的根，则可以使用多个后缀。

默认情况下，标准目录服务器部署包含多个后缀，一个用于存储数据，另一个用于存储内部目录操作（如配置信息和目录模式）所需的数据。有关这些标准目录后缀的更多信息，请参阅 *红帽目录服务器管理指南*。

4.2.1.1. 后缀命名

目录中的所有条目都应位于一个通用的基础条目下，*根后缀*。当为根目录后缀选择名称时，请考虑这四个点以使名称有效：

- 全局唯一。
- 静态，因此很少会在发生改变的情况下。
- 简而言之，这样，在屏幕上更轻松地阅读条目。
- 易于输入并记住。

在单一企业环境中，选择一个与企业 DNS 名称或互联网域名一致的目录后缀。例如，如果企业拥有 **example.com** 的域名，则目录后缀在逻辑上是 **dc=example,dc=com**。

dc 属性通过将域名拆分到其组件部分来代表后缀。

通常，任何属性都可用于命名根后缀。但是，对于托管机构，将 root 后缀限制为以下属性：

- ***dc*** 定义域名的组件。
- ***c*** 包含代表国家名称的双位代码，由 ISO 定义。
- ***l*** 标识条目所在的数目、城市或其他地理位置，或者与该条目相关联。
- ***st*** 识别条目所在的状态或省去。
- ***o*** 标识条目所属组织的名称。

这些属性的存在允许与订阅者应用程序进行互操作性。例如，托管机构可能会使用这些属性为其其中一个客户端创建根后缀，如 **o= example_a、st=Washington,c=US**。

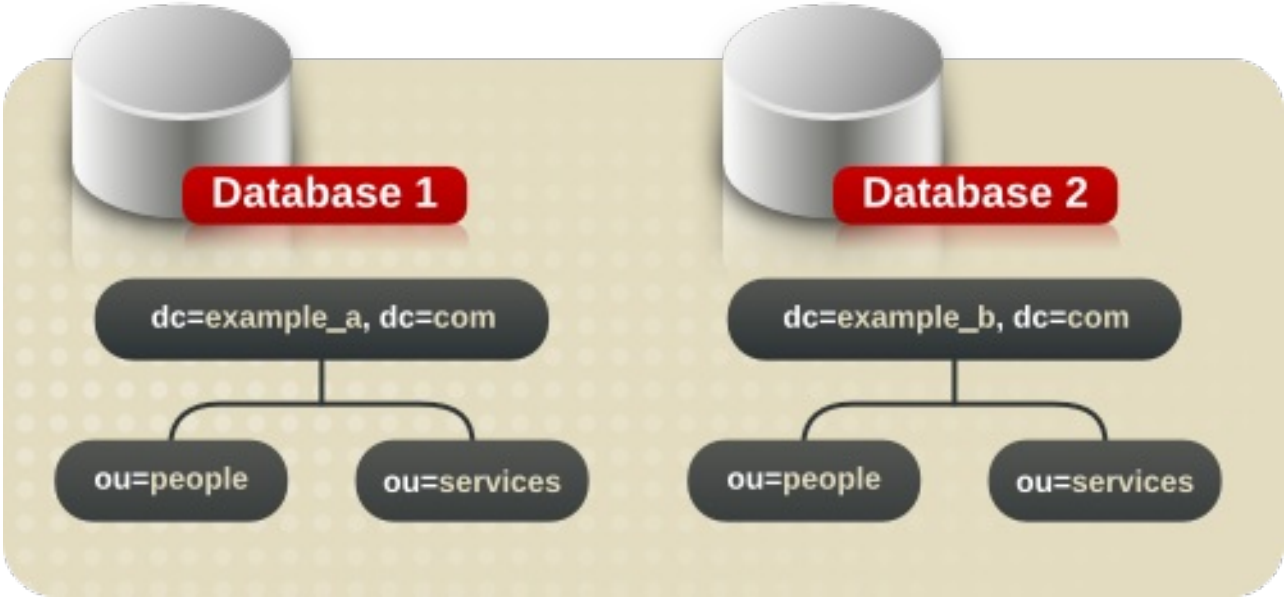
使用机构名称后加上国家设计是后缀的典型 X.500 命名约定。

4.2.1.2. 命名多个 Suffixes

与目录搭配使用的每个后缀都是唯一的目录树。可以通过多种方式在目录服务中包含多个树。第一种方法是创建存储在由 Directory Server 提供的独立数据库中的多个目录树。

例如，为 **example_a** 和 **example_b** 创建单独的后缀，并将它们存储在单独的数据库中。

图 4.1. 在数据库中包含多个目录树



根据资源限制，数据库可以存储在单一服务器或多个服务器上。

4.2.2. 创建目录树结构

决定是否使用平面或分级树结构。作为常规规则，尝试尽可能使目录树变为扁平。但是，以后在多个数据库间分区信息、准备复制或设置访问控制时，一定程度的层次结构非常重要。

树结构涉及以下步骤和注意事项：

- 第 4.2.2.1 节 “对目录进行分支”
- 第 4.2.2.2 节 “识别分支点”
- 第 4.2.2.3 节 “复制注意事项”
- 第 4.2.2.4 节 “访问控制注意事项”

4.2.2.1. 对目录进行分支

设计层次结构以避免有问题的名称更改。命名空间扁平化，名称不太可能改变。名称更改的可能性与名称中可能更改的组件数量大致成比例。更容易对目录树、名称中的更多组件以及名称更有可能更改的等级。

以下是设计目录树层次结构的一些准则：

- 将树分支，仅代表企业中最大的组织细分。

  所有这些分支点应限制在部门（如公司信息服务、客户支持、销售与工程）。确保用于分支目录树的部门具有稳定的；如果企业经常重新组织，则不执行此类分支。

- 对分支点使用功能或通用名称而不是实际的机构名称。

  名称更改。虽然子树可以重命名，但它是具有许多子条目的大型后缀的很长和资源密集型过程。使用代表机构功能的通用名称（例如，使用 **Engineering** 而不是 **Widget Research 和 Development**）使得在机构或项目更改后需要重命名子树的几率要小。

- 如果有多个机构执行类似的功能，请尝试为该功能创建单个分支点，而不是基于划分行进行分支。

例如，即使有多个营销机构，每个营销机构都负责特定的产品行，请创建一个 **ou=Marketing** 子树。然后，所有营销条目都属于该树。

Enterprise 环境中的分支

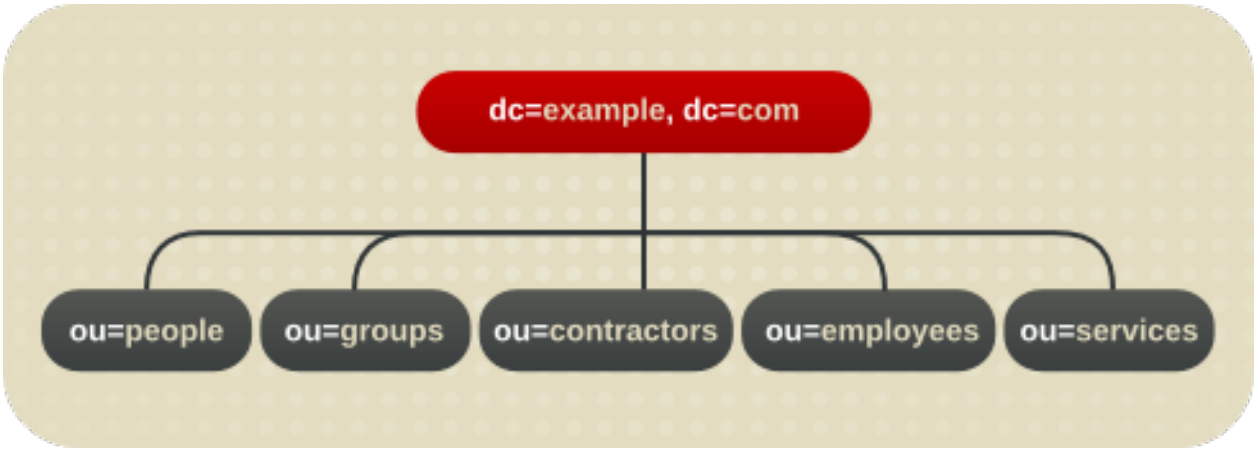
如果目录树结构基于可能更改的信息，可以避免名称更改。例如，对树中的对象类型而非组织，对对象类型的结构为基础。这有助于避免在组织单元之间影响一个条目，这需要修改可分辨名称 (DN)，这是代价昂贵的操作。

有几种通用对象可用于定义结构：

- **ou=people**
- **ou=groups**
- **ou=services**

使用这些对象组织组织的目录树可能如下所示。

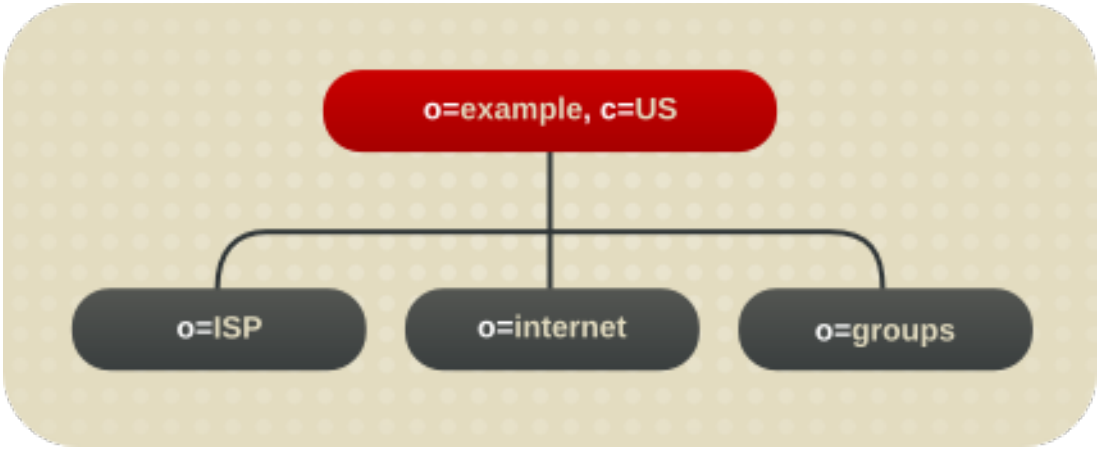
图 4.2. Environment Directory 树示例



主机环境中的分支

对于托管环境，创建一个树，其中包含对象类 *组织* (***o***)的两个条目，以及对象类 ***organizationalUnit*** (***ou***) 下的一个条目。例如，ISP 分支其目录示例，如下所示。

图 4.3. 主机目录树示例



4.2.2.2. 识别分支点

在规划目录树中的分支时，决定使用什么属性来识别分支点。请记住，DN 是由属性数据对组成的唯一字符串。例如： Barbara Jensen (Example Corp. 的员工)条目的 DN 是 **uid=bjensen,ou=people,dc=example,dc=com**。

每个属性对代表目录树中的一个分支点，如企业 Example Corp 的目录树中所示。图 4.4 “示例 Corp 的 Directory 树。”

图 4.4. 示例 Corp 的 Directory 树。

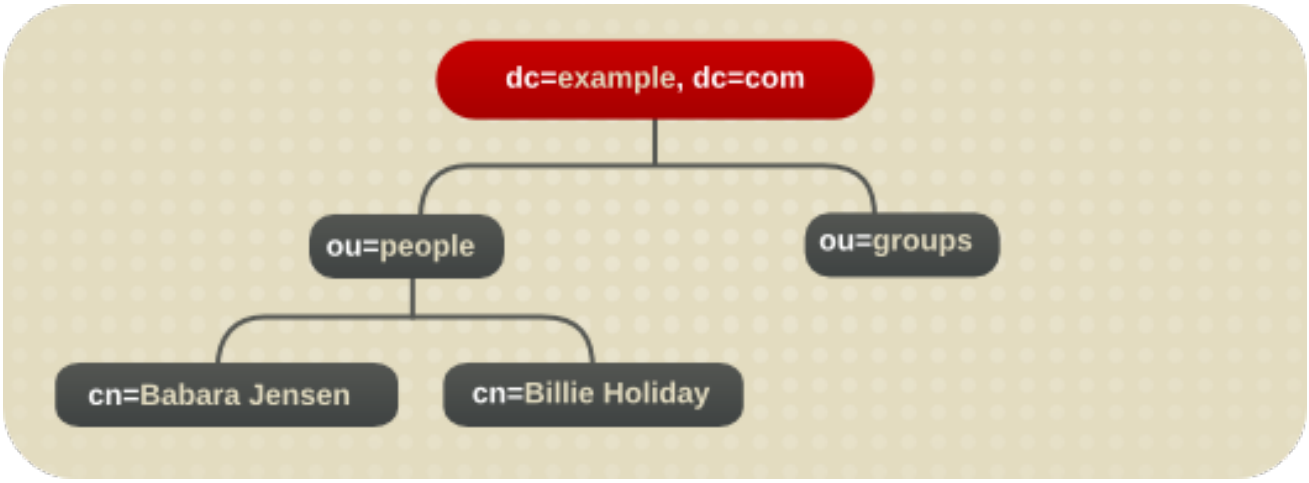
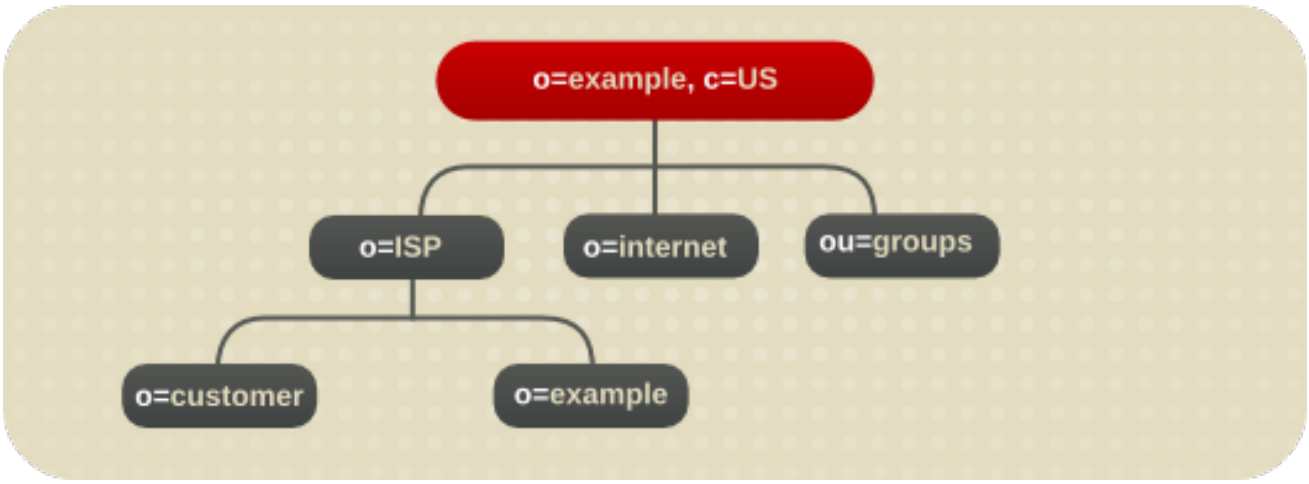


图 4.5 “示例 ISP 的目录树” 显示 ISP （互联网主机）的目录树。

图 4.5. 示例 ISP 的目录树



在后缀条目 **c=US,o=example** 下，树被分成三个分支。ISP 分支包含客户数据和 ISP 示例内部信息。互联网分支是域树。groups 分支包含有关管理组的信息。

在为分支点选择属性时请考虑以下几点：

- 保持一致。

  如果区分名称 (DN) 格式在目录树中不一致，则一些 LDAP 客户端应用程序可能会混淆。也就是说，如果 ***l*** 属于目录树的一个部分的 ***ou***，请确保在目录服务的所有其他部分的 ***l*** 从属为 ***ou***。

- 尝试只使用传统属性（在 第 4.2.2.2 节 “识别分支点”中显示）。

  使用传统属性会增加保留与第三方 LDAP 客户端应用程序的兼容性的可能性。使用传统属性还意味着它们对默认目录架构所知，这有助于为分支 DN 构建条目。

表 4.1. 传统 DN 分支点属性

属性	定义
dc	域名的一个元素，如 **dc=example** ；这经常在对中指定，甚至更长，具体取决于域，如 **dc=example,dc=com** 或 **dc=mtv,dc=example,dc=com**。
c	国家/地区名称。
o	机构名称。此属性通常用于代表一个大的分部分支，如公司部门、学术线（人类、科学）、子公司，或者企业内的其他主要分支，如 第 4.2.1.1 节 “后缀命名”。
ou	组织单元。此属性通常用来代表比组织较小的部门分支。组织单元一般与上述机构从属。
st	状态或省名称。
l 或 ***locality***	本地性，如城市、国家、办公室或设备名称。

注意

常见错误是假定根据可分辨名称中使用的属性搜索目录。区分名称只是目录条目的唯一标识符，不能用作搜索键。相反，请根据条目本身中存储的属性对搜索条目。因此，如果条目的可分辨名称为 **uid=bjensen,ou=People,dc=example,dc=com**，则搜索 **dc=example** 不匹配该条目，除非在该条目中明确添加了 **dc:example** 作为属性。

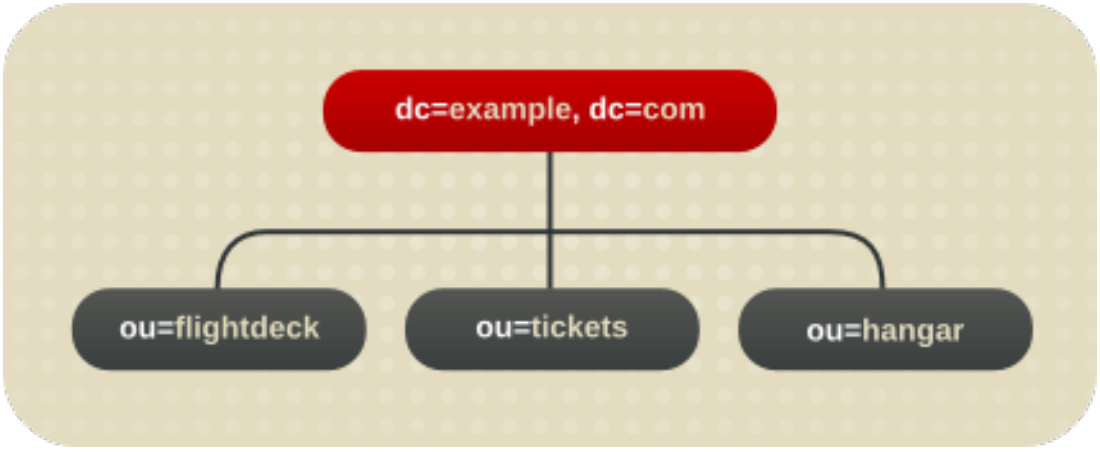
4.2.2.3. 复制注意事项

在目录树设计过程中，请考虑要复制哪些条目。描述要复制的一组条目的自然方法是指定子树顶部的 DN，并复制其下面的所有条目。此子树也与数据库对应，它是包含部分目录数据的目录分区。

例如，在企业环境中，一种方法是组织目录树，使其与企业中的网络名称对应。网络名称不会更改，因此目录树结构是稳定的。此外，使用网络名称创建目录树的顶层分支很有用，在使用复制来将复制绑定到不同的目录服务器时很有用。

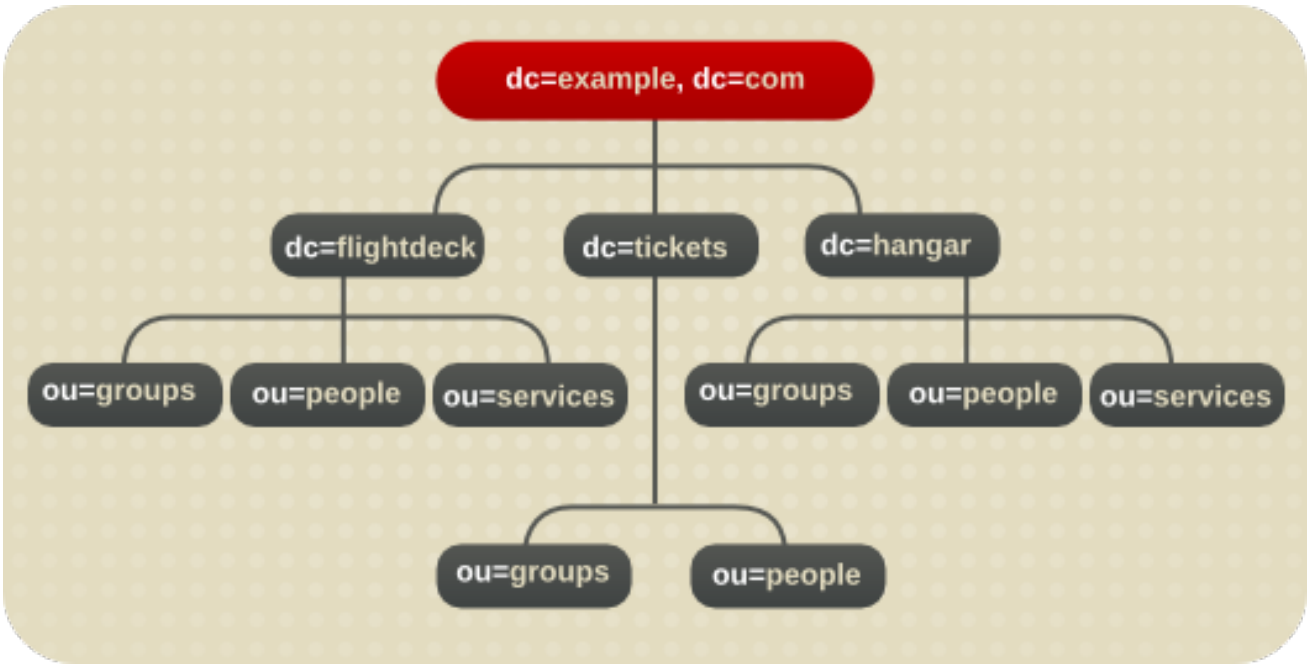
例如，Example Corp. 有三个主要网络，称为 **flightdeck.example.com**、**tickets.example.com** 和 **hangar.example.com**。它们最初将其目录树分到其主要组织部门的三个主要组中。

图 4.6. 示例公司的 Directory 树的初始分支。



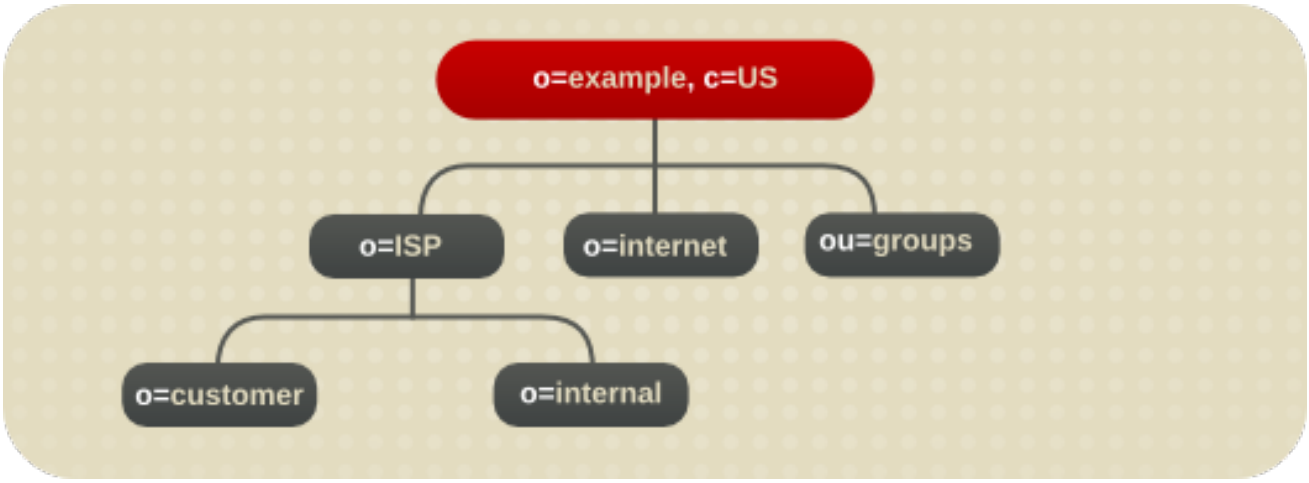
在创建了树的初始结构后，他们会创建额外的分支来显示每个机构组的分类。

图 4.7. 为示例公司扩展分支.



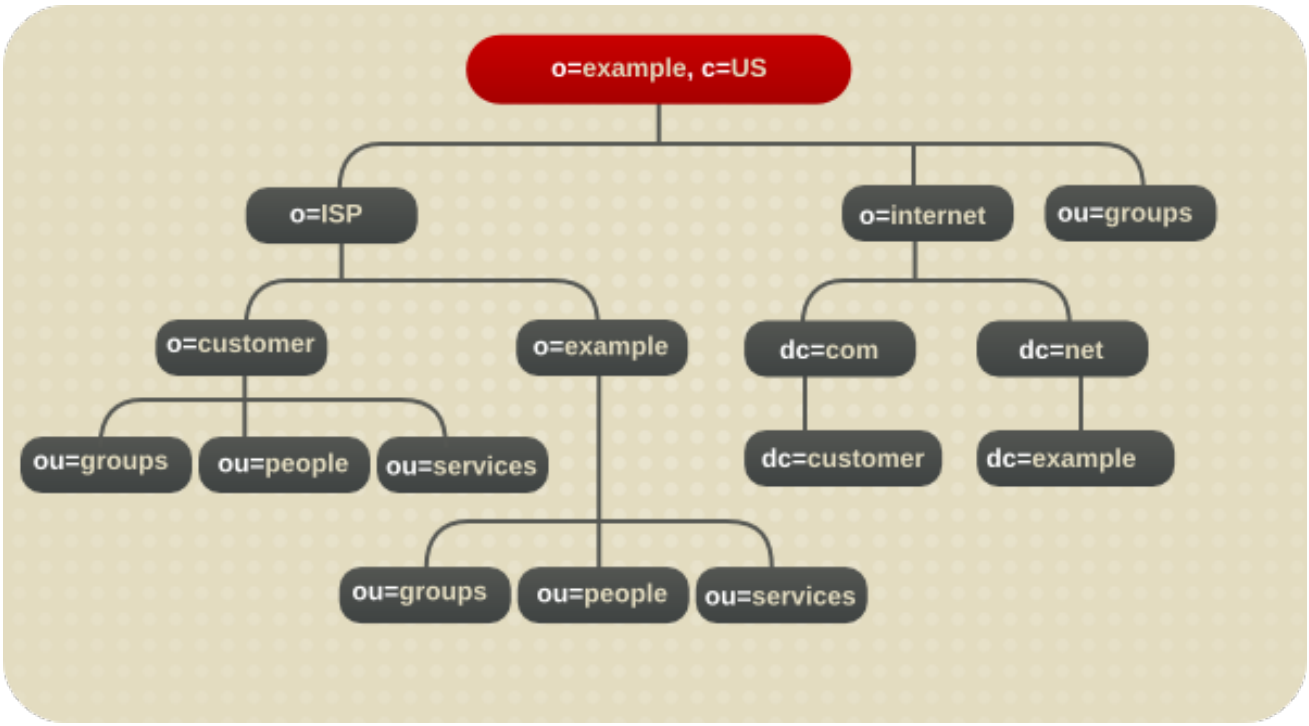
示例 ISP 在非对称树中对目录进行镜像，以镜像其组织的非对称树进行分支。

图 4.8. 示例 ISP 的目录分支



在创建了其目录树的初始结构后，会为逻辑子组创建额外的分支。

图 4.9. 用于示例 ISP 的扩展分支



企业和托管组织均基于可能经常更改的信息设计其数据层次结构。

4.2.2.4. 访问控制注意事项

在目录树中引入层次结构，以启用某些类型的访问控制。与复制一样，可以更轻松地对类似的条目进行分组，然后从单个分支进行管理。

也可以通过分级目录树启用管理分布。例如，为管理员授予营销部门对营销条目的访问权限，以及来自销售部门对销售条目的管理员访问，根据这些部门对目录树进行设计。

访问控制可以基于目录的内容而不是目录树。过滤的机制可以定义一个访问控制规则，说明目录条目可以访问包含特定属性值的所有条目。例如，设置一个 ACI 过滤器，向 sales 管理员提供对包含属性值 **ou=Sales** 的所有条目的访问权限。

但是 ACI 过滤器难以管理。确定哪个访问控制方法最适合于 目录：组织树层次结构中的分支、ACI 过滤器或两者的组合。

4.2.3. 命名条目

在设计目录树的层次结构后，决定在命名结构中的条目时要使用的属性。通常，通过选择一个或多个属性值来形成名称，以形成 *相对可分名称(RDN)*。RDN 是 DN 中的单个组件。这是显示的第一个组件，因此该组件的 属性是 *naming 属性*，因为它为该条目设置唯一名称。要使用的属性取决于被命名的条目类型。

条目名称应遵循以下规则：

- 为命名选择的属性应不太可能改变。
- 该名称在目录中必须是唯一的。

  唯一名称可确保 DN 在 目录中最多可以看到一个条目。

在创建条目时，在条目中定义 RDN。通过在该条目中定义至少 RDN，该条目可以更轻松地找到该条目。这是因为搜索不会针对实际 DN 执行，而是在条目本身中存储的属性值。

属性名称具有含义，因此尝试使用与它所代表的输入类型匹配的属性名称。例如，不要使用 ***l*** 来代表一个机构，或者 ***c*** 代表组织单元。

- 第 4.2.3.1 节 “命名角色条目”
- 第 4.2.3.2 节 “命名组条目”
- 第 4.2.3.3 节 “命名机构条目”
- 第 4.2.3.4 节 “命名其他 Entries 的 Kind”

4.2.3.1. 命名角色条目

该用户条目的名称(DN)必须是唯一的。通常，区分名称使用 ***commonName*** 或 ***cn*** 属性来命名其个人条目。也就是说，名为 Babs Jensen 的个人可能具有可区分名称 **cn=Babs Jensen,dc=example,dc=com** 的条目。

虽然使用通用名称可以更容易地将人与条目相关联，但可能不足以排除具有相同名称的人。这很快会导致一个称为 *DN 名称冲突* 的问题，多个条目具有相同的可辨识名称。

通过在通用名称中添加唯一标识符来避免常见的名称冲突，如 **cn=Babs Jensen+employeeNumber=23,dc=example,dc=com**。

但是，这可以导致大型目录的通用名称被放大，并且难以维护。

更好的方法是识别带有 ***cn*** 以外的某些属性的 person 条目。考虑使用以下属性之一：

- ***uid***

  使用 ***uid*** 属性指定个人的一些唯一值。可能包括用户登录 ID 或员工号码。托管环境中的订阅者应该由 ***uid*** 属性识别。

- ***mail***

  mail 属性包含个人的电子邮件地址，该地址始终是唯一的。这个选项可能会导致包含重复属性值（如 **mail=bjensen@example.com,dc=example,dc=com**）的 awkward DNs，因此仅在没有与 ***uid*** 属性一起使用的其他唯一值时才使用这个选项。例如，如果企业没有为临时或合同员工分配员工数量或用户 ID，则使用 ***mail*** 属性而不是 ***uid*** 属性。

- ***employeeNumber***

  对于 **inetOrgPerson** 对象类的员工，请考虑使用人员分配的属性值，如 ***employeeNumber***。

任何用于个人条目 RDN 的属性数据对，请确保它们是唯一的永久值。人员条目还应是可读的。例如，**uid=bjensen,dc=example,dc=com** 可用于 **uid=b12r56A,dc=example,dc=com**，因为可识别的 DN 简化了某些目录任务，如根据可分辨的名称更改目录条目。另外，一些目录客户端应用程序假定 ***uid*** 和 ***cn*** 属性使用人类可读的名称。

在托管环境中为 Person 条目的注意事项

如果一个人是服务的订阅者，则条目应该是对象类 **inetUser**，条目应包含 ***uid*** 属性。在客户子树中，属性必须是唯一的。

如果个人是托管机构的一部分，请将它们表示为带有 **nsManagedPerson** 对象类的 **inetOrgPerson**。

将 Person Entries 放置到 DIT 中

以下是将人员条目放入目录树的一些准则：

- 企业中的人员应位于组织条目下的目录树中。
- 对托管机构的订阅者需要低于托管机构的 **ou=people** 分支。

4.2.3.2. 命名组条目

有四个主要代表组的方法：

- *静态组* 显式定义是成员。**groupOfNames** 或 **groupOfUniqueNames** 对象类包含命名组成员的值。静态组适合具有几个成员的组，如目录管理员组。静态组不适用于具有数千个成员的组。

  静态组条目必须包含 **uniqueMember** 属性值，因为 **uniqueMember** 是 **groupOfUniqueNames** 对象的强制属性。此对象类需要 ***cn*** 属性，它可用于组成组条目的 DN。

- *动态组* 使用一个代表搜索过滤器和子树的组的条目。与过滤器匹配的条目是组的成员。
- *角色* 统一了静态和动态组概念。如需更多信息，请参阅 第 4.3 节 “分组目录条目”。

在包含托管机构的部署中，请考虑使用 **groupOfUniqueNames** 对象类包含命名目录管理中使用的组成员的值。在托管机构中，我们还建议用于目录管理的组条目位于 **ou=Groups** 分支下。

4.2.3.3. 命名机构条目

与其他条目名称一样，组织条目名称必须是唯一的。将机构的法律名称与其他属性值一起使用有助于确保名称是唯一的，如 **o=example_a+st=Washington,o=ISP,c=US**。

也可以使用商标，但不能保证其唯一。

在托管环境中，使用 ***organization*** (***o***)属性作为 naming 属性。

4.2.3.4. 命名其他 Entries 的 Kind

该目录包含包括许多内容的条目，如本地城市、州、国家、设备、服务器、网络信息和其他种类的数据。

对于这些类型的条目，请在 RDN 中使用 ***cn*** 属性（如果可能）。然后，为命名组条目，将其命名为 **cn=administrators,dc=example,dc=com** 等内容。

但是，有时条目的对象类不支持 ***commonName*** 属性。反之，使用条目对象类支持的属性。

用于条目 DN 的属性不必与条目中实际使用的属性对应。但是，在 DN 属性与条目使用的 DN 属性之间有一些关联，可以简化目录树的管理。

4.2.4. 重命名条目和子树

第 4.2.3 节 “命名条目” 讨论红帽目录服务器中命名条目的重要性。条目名称在某种意义上定义目录树结构。每个分支点（其下方具有条目的每个条目）会在层次结构中创建新链接。

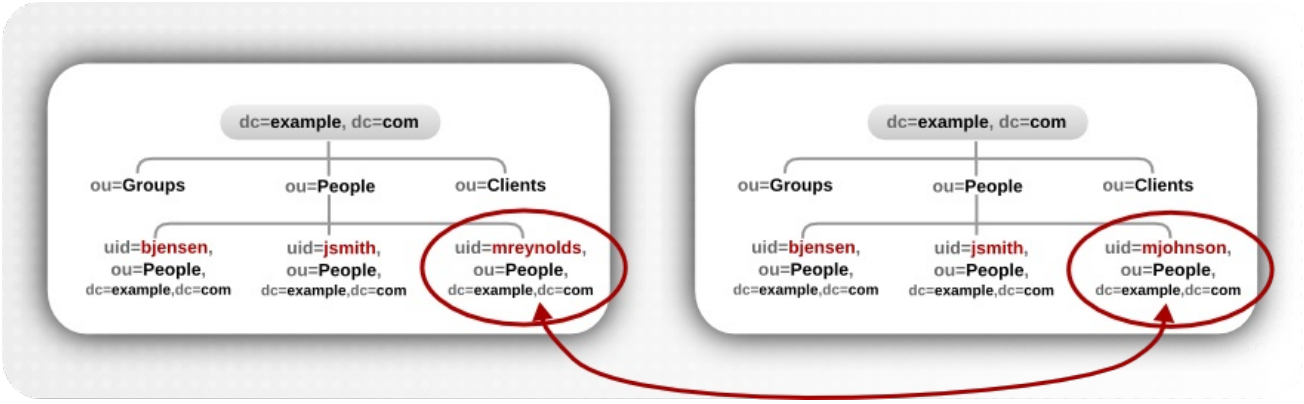
例 4.1. building Entry DNs

```
dc=example,dc=com  =>  root suffix
ou=People,dc=example,dc=com  =>  org unit
st=California,ou=People,dc=example,dc=com  =>  state/province
l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  city
ou=Engineering,l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  org
```

unit
uid=jsmith,ou=Engineering,l=Mountain View,st=California,ou=People,dc=example,dc=com => leaf entry

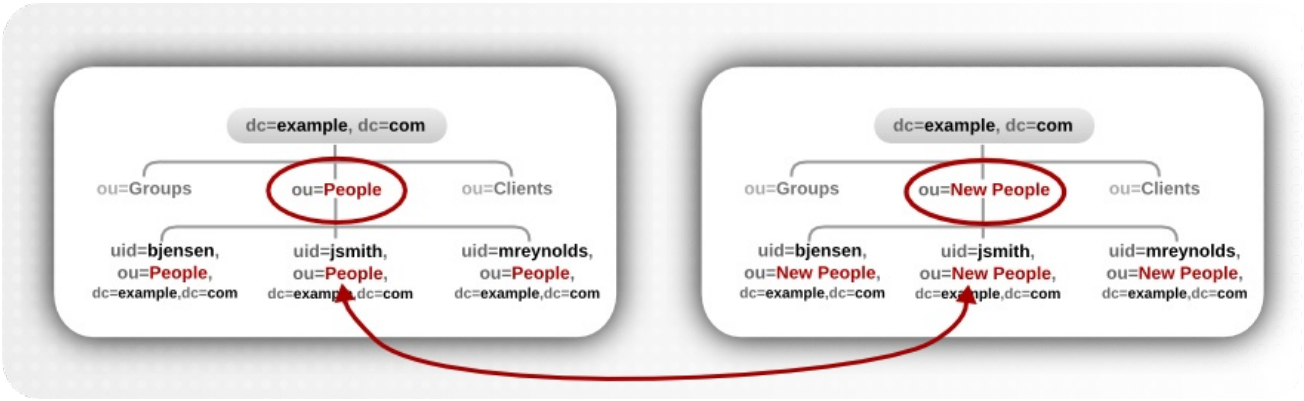
当条目的 naming 属性时，DN 的左边元素已更改，这是 *modrdn 操作*。这是一种特殊的修改操作，因为在某种意义上，它会将条目移到目录树中。对于叶条目（没有子项的项），modrdn 操作是侧向移动的；条目具有相同的父项，只需一个新名称。

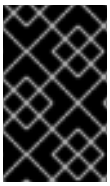
图 4.10. Leaf Entry 的 modrdn Operations



对于子树条目，modrdn 操作不仅重命名子树条目本身，还会更改子树 下下所有子条目的 DN 组件。

图 4.11. 子树条目的 modrdn 操作



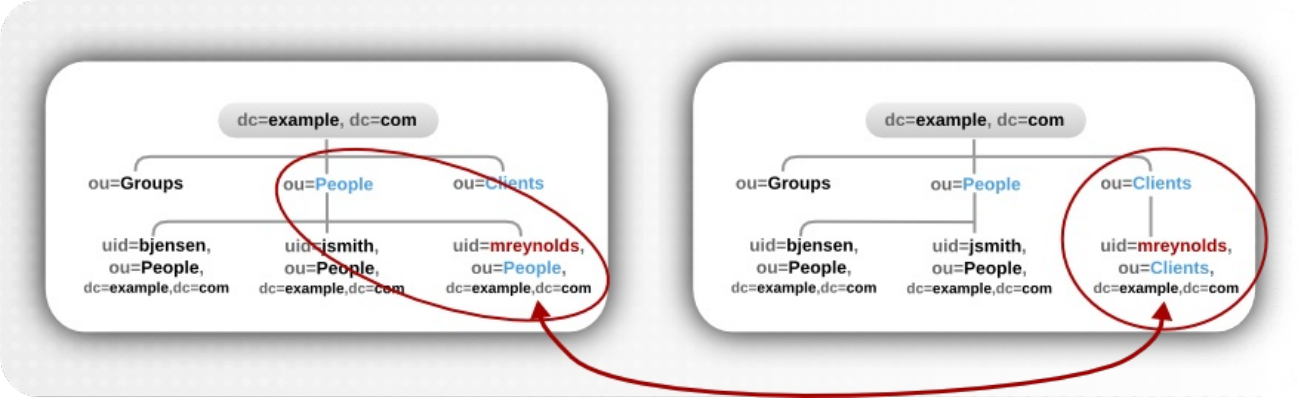


重要

子树 modrdn 操作也会移动并重命名 subtree 条目下的所有子条目。对于大型子树，这可以是一个时间和资源密集型进程。以层次结构来规划对目录数的命名结构，使其不需要频繁对子树进行重命名操作。

重命名子树的类似操作是将条目从一个子树移到另一个子树。这是一个扩展的 modrdn 操作类型，它同时重命名条目（即使是相同的名称），并且设置 ***newsuperior*** 属性，它将条目从一个父项移到另一个父项。

图 4.12. 对一个新父项的 modrdn 操作



新的 superior 和 subtree rename 操作都是可能的，因为条目如何存储在 **entryrdn.db** 索引中。每个条目都由其自己的键（一个 *self-link*）标识，然后是一个子键来标识其父项（ *父链接*）和任何子项。这是一个层次结构目录树的格式，它将父项和子项视为条目的属性，每个条目都由唯一 ID 及其 RDN 描述，而不是完整的 DN 进行标识。

```
numeric_id:RDN => self link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
P#:RDN => parent link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
C#:RDN => child link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
```

例如，**ou=people** 子树的父项为 **dc=example,dc=com**，子是 **uid=jsmith**。

```
4:ou=people
  ID: 4; RDN: "ou=People"; NRDN: "ou=people"
P4:ou=people
  ID: 1; RDN: "dc=example,dc=com"; NRDN: "dc=example,dc=com"
C4:ou=people
  ID: 10; RDN: "uid=jsmith"; NRDN: "uid=jsmith"
```

在执行重命名操作时需要记住以下一些事项：

- 您无法重命名 root 后缀。
- 子树重命名操作对复制的影响最少。复制协议会应用于整个数据库，而不是数据库中的子树，因此子树重命名操作不需要重新配置复制协议。子树重命名操作之后的所有名称都会正常进行复制。
- 重命名子树 **可能需要** 重新配置任何同步协议。同步协议在后缀或子树级别上设置，因此重命名子树可能会破坏同步。
- 重命名子树 **要求** 手动重新配置子树级 ACI，以及为子树子条目设置的任何条目级 ACI。
- 您可以将子树重命名为子树，但您无法删除子树。
- 尝试更改子树的组件，如从 ***ou*** 移到 ***dc***，可能会因为 schema 违反而失败。例如，**organizationalUnit** 对象类需要 ***ou*** 属性。如果该属性作为重命名子树的一部分被删除，则操作将失败。

4.3. 分组目录条目

创建所需条目后，对其进行分组以便便于管理。Directory 服务器支持多种分组条目的方法：

- 使用组
- 使用角色

4.3.1. 关于组

组（如名称所示）只是用户集合。Directory 服务器中有几个不同类型的组，它反映了允许的成员资格类型，如证书组、URL 组和唯一组（每个成员都必须是唯一的）。每种组类型都由对象类（如 **groupOfUniqueNames**）和对应的 member 属性（如 **uniqueMember**）定义。

组的类型标识成员的类型。组的配置取决于这些成员如何添加到组中。目录服务器有两种组：

- *静态组* 有有限且定义的成员列表，这些成员被手动添加到组条目中。
- *动态组*使用过滤器来识别哪些条目是组的成员，因此当与组过滤器更改的条目时，组成员资格将不断改变。

组是目录服务器中组织条目的最简单形式。它们主要是手动配置的，除组织方法之外没有功能或行为。（通常，组"不会"对目录条目进行任何操作，尽管组可以被 LDAP 客户端使用来执行操作。）

4.3.1.1. 列出用户条目中的组成员资格

组本质上就是用户 DN 的列表。默认情况下，组成员资格只会反映在组条目本身中，而不反映在用户条目中。但是，MemberOf 插件使用组成员条目动态更新 *用户条目*，以反映用户所属的组。MemberOf 插件自动扫描具有指定 member 属性的组条目，跟踪所有用户 DN，并在用户条目上创建对应的 ***memberOf*** 属性，其名称为组的名称。

组成员资格由组条目上的 member 属性 *决定*，但用户的所有组的组成员资格 *反映在* ***memberOf*** 属性中的用户条目中。用户所属的每个组的名称列为 ***memberOf*** 属性。这些 ***memberOf*** 属性的值由 Directory 服务器管理。

注意

如 第 6.2.1 节 "关于使用多个数据库" 中所述，可以将不同的后缀存储在不同的数据库中。

默认情况下，MemberOf 插件只查找与组相同的数据库中的用户的潜在成员。如果用户存储在与组不同的数据库中，则不会使用 ***memberOf*** 属性更新用户条目，因为插件无法确定它们之间的关系。

MemberOf 插件可以通过启用 ***memberOfAllBackends*** 属性配置为搜索所有配置的数据库。

在插件条目中设置 multi-valued ***memberofgroupattr***，可以将 MemberOf 插件的单一实例配置为识别多个成员属性，以便 MemberOf 插件可以管理多种类型的组。

4.3.1.2. 自动添加新条目到组

组管理是管理目录数据的一个关键因素，特别是对于使用 Directory Server 数据和机构或者组将功能应用到条目的客户端。组可以更轻松地在目录中一致、可靠地应用策略。密码策略、访问控制列表和其他规则都可以基于组成员资格。

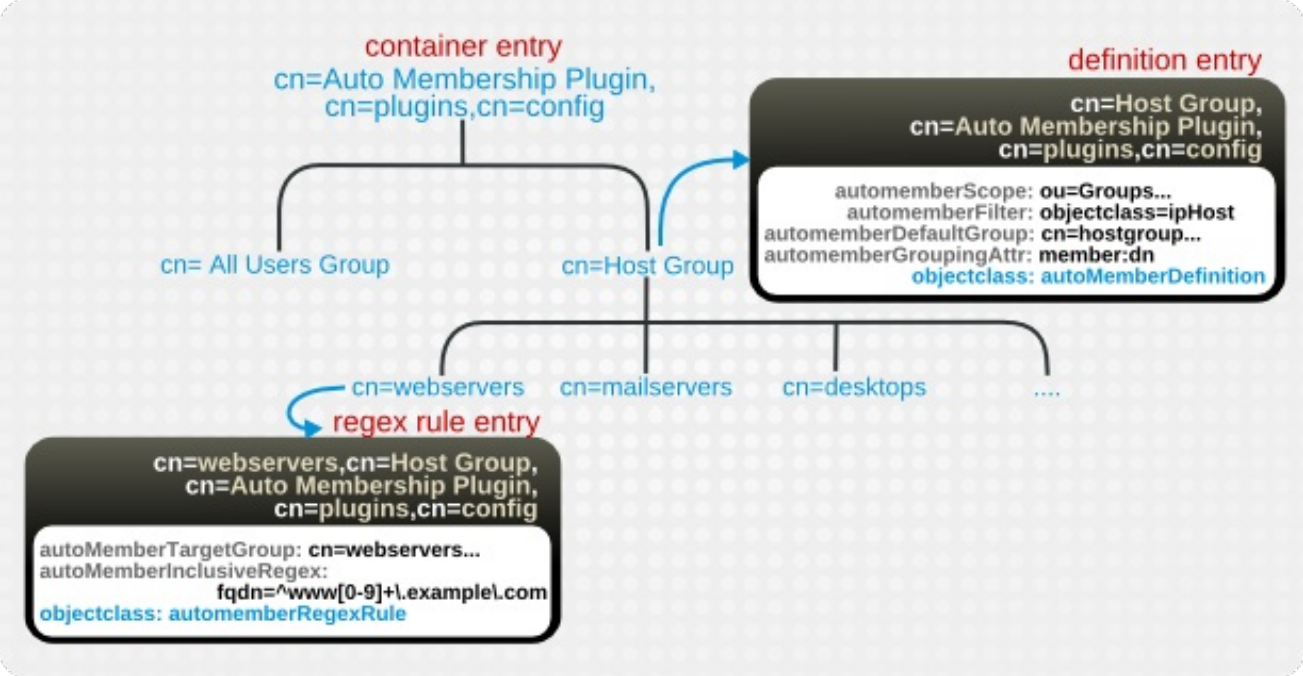
能够在创建帐户时自动将新条目分配给组，确保适当的策略和功能会立即应用到这些条目 - 无需管理员干预。

Automembership 插件 本质上允许静态组充当动态组。它使用一组规则（基于条目属性、目录位置和正则表达式）将用户自动分配给指定组。

根据某些其他属性的值，可能存在与 LDAP 搜索过滤器匹配的实例条目。例如，可能需要根据机器的 IP 地址或物理位置将计算机添加到不同的组中；用户可能需要根据员工的 ID 号而位于不同的组中。

automember 定义是一组嵌套条目，以及 Auto Membership 插件容器，然后自动成员定义，然后是该定义的任何正则表达式条件。

图 4.13. 正则表达式条件



注意

只有在将某个条目添加到目录服务器时，才会自动进行自动分配。

对于为满足自动成员规则而编辑的现有条目或条目，有一个可运行的修复任务来分配正确的组成员资格。

4.3.2. 关于角色

角色（role） 是一组混合组，可作为静态和动态组。在使用组时，条目作为成员添加到组条目中。通过角色，role 属性添加到条目中，然后使用该属性自动识别角色条目中的成员。

角色 *以多种不同方式有效且自动* 组织用户：

- *明确列出角色成员*。查看角色将显示该角色的完整列表。角色本身可以查询以检查成员资格（无法通过动态组）。
- *显示条目所属角色*。因为角色成员资格由条目上的属性决定，只需查看条目即可显示它所属的所有角色。这和组的 *memberOf* 属性类似，只需要启用或配置插件实例才能正常工作。它是自动的。
- *分配适当的角色*。角色成员资格通过 *条目（entry）* 进行分配，而不是通过角色分配，因此单个步骤中可以通过编辑条目轻松地分配和移除用户所属的角色。

受管角色可以执行通常可通过静态组完成的所有操作。角色成员可以使用过滤的角色过滤，与使用动态组的过滤类似。角色比组更容易使用，在实施中更灵活，并降低客户端复杂性。

角色 *成员* 是拥有角色的条目。成员可以显式指定或动态指定。指定角色成员资格的方式取决于角色的类型。目录服务器支持三种类型的角色：

- *受管角色* 具有明确的 enumerated 列表。
- 根据 *每个条目包含的属性（在 LDAP 过滤器中指定）*，*过滤* 过滤角色将条目分配给角色。与过滤器匹配的条目拥有角色。
- *嵌套角色*是包含其他角色的角色。

角色。通过激活/取消激活角色的概念，允许激活整个组或者仅激活一个操作。例如，可以通过取消激活角色所属的角色来暂时禁用角色。

当某个角色处于不激活状态时，这并不表示用户无法绑定到使用那个角色条目的服务器。inactivated 角色的含义是，用户无法使用属于该角色的任何条目绑定到服务器；属于 inactivated 角色的条目会将 ***nsAccountLock*** 属性设置为 **true**。

当嵌套角色处于激活时，如果用户在嵌套角色内是任何角色的成员，则无法绑定到服务器。所有属于直接或间接属于角色的条目都是嵌套角色的成员，将 ***nsAccountLock*** 设置为 **true**。可以有多个嵌套角色层，并在嵌套的任意点上激活嵌套角色，则会在它下面的所有角色和用户中激活。

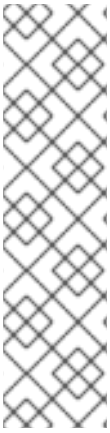
4.3.3. 确定角色和组之间的选择

角色和组群可以完成相同的目标。受管角色可以执行静态组可以执行的操作，而过滤后的角色也可过滤和将成员识别为动态组执行的操作。角色和组都各有优缺点。决定使用角色或组（或混合）依赖于平衡客户端需求和服务器资源。

角色可以降低客户端的复杂性，这是他们的关键优势。使用角色时，客户端应用程序可以通过搜索条目上的 ***nsRole*** 操作属性来检查角色成员资格；此多值属性标识条目所属的每个角色。从客户端应用程序视图中，检查成员资格的方法统一并在服务器端执行。

然而，对于客户端而言，这种易用性会增加服务器复杂性。 与评估组相比，评估角色对目录服务器的资源密集型，因为服务器能够为客户端应用工作。

虽然组对于服务器更容易，但它们需要更智能、更复杂的客户端来有效地使用它们。例如，从应用程序角度而言，动态组不提供服务器中的支持，以提供组成员列表。相反，应用会检索组定义，然后运行过滤器。只有在配置了适当的插件时，组成员资格才会反映在用户条目中。最终，决定组成员资格无法统一或可预测的方法。



注意

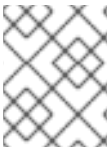
可平衡组成员资格管理组成员的一个方面是 MemberOf 插件。使用 ***memberOf*** strikes a nice 平衡，使客户端使用且对服务器进行计算效率。

当用户添加到组中时，MemberOf 插件会在用户条目上动态创建 ***memberOf*** 属性。客户端可以在组条目上运行单个搜索，以获取其所有成员的列表，或者在用户条目上搜索单个搜索来获取它所属的所有组的完整列表。

只有修改成员资格时，服务器才会有维护开销。由于指定的成员(group)和 ***memberOf*** (user)属性都存储在数据库中，因此不需要额外的处理来搜索，从而使从客户端搜索非常高效。

4.4. 虚拟目录信息树视图

目录服务器支持分层导航和组织（称为 *虚拟目录信息树视图* 或 *虚拟 DIT* 视图）的概念。



注意

虚拟视图与多个后端不兼容，因为视图返回的条目必须位于相同的后端中；搜索范围仅限于一个后端。

4.4.1. 关于虚拟 DIT 视图

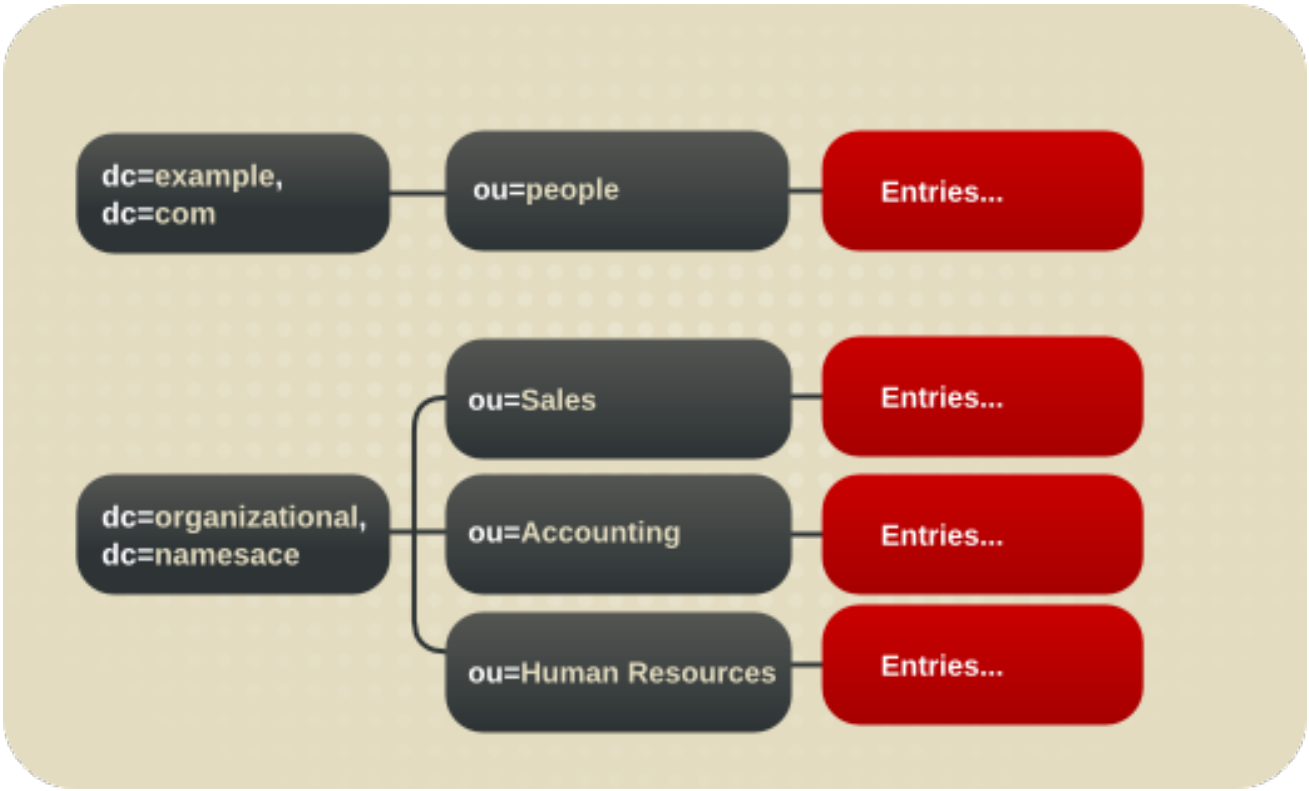
配置目录命名空间的方法有两种：

- 分层目录信息树。
- 扁平目录信息树。

分层 DIT 可用于浏览目录，但过于繁琐且易更改。对分层 DIT 的主要变化可能比较昂贵且耗时的操作，因为它通常涉及显著的服务中断。这通常只能通过在小时和低流量期间执行更改来最小化。

在不需要更改的情况下，扁平 DIT 不会提供便捷的方法来浏览或管理目录服务中的条目。扁平化 DIT 还带来了许多管理挑战，因为管理变得更为复杂，无需任何自然的分级分组。

图 4.14. Flat 和组织的 DIT 示例



使用分级 DIT 时，部署必须确定层次结构的主体域。只能做出一个选择；自然倾向于选择组织层次结构。

在很多情况下，组织的这种视图适合良好，但只有一个视图可能非常限制用于目录导航和管理。例如，组织层次结构非常适用于查找属于 Accounts 部门的人员的条目。但是，对于查找属于地理位置的用户（如 Mountain View, California）的条目，这个视图非常有用。第二个查询在第一个查询一样有效，但需要了解条目中包含的属性和其他搜索工具。在这种情况下，使用 DIT 进行导航不是选项。

同样，当 DIT 符合管理功能的要求时，管理该目录将更加容易。DIT 的机构也可能受到其他因素的影响，如复制和迁移注意事项，这会导致 DIT 能够对这些应用程序具有功能，但其他情况下的实际实用程序很少。

显然，层次结构是导航和管理的一个非常有用的机制。不过，为了避免对现有 DIT 进行更改的负担，部署可能会完全默认使用扁平 DIT。

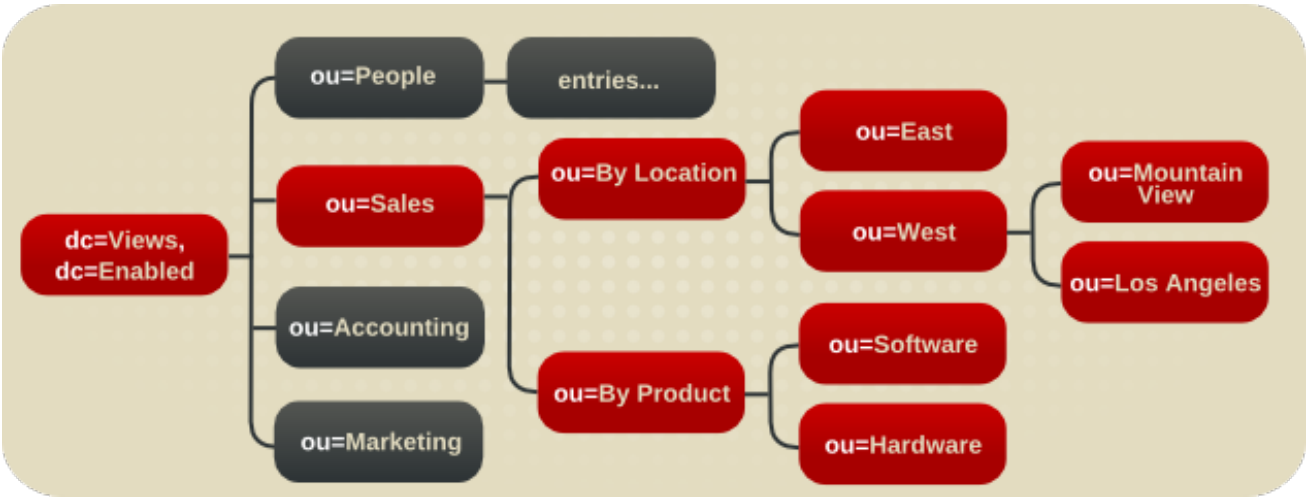
如果某个目录提供了创建任意数量的层次结构，则部署具有优势，不必将目标条目移到条目上，而无需移动问题的目标条目。Directory 服务器的 *虚拟 DIT 视图* 功能解决了决定用于目录部署的 DIT 类型的定性。

虚拟 DIT 视图 提供了一种对条目进行分层导航的方法，而无需在任何特定位置存在这些条目。虚拟 DIT 视图使用有关条目的信息将其放在视图层次结构中。对于客户端应用，虚拟 DIT 视图显示为普通容器层次结构。在某种意义上，虚拟 DIT 视图对一组条目具有超级 DIT 层次结构，与这些条目无关，无论这些条目是否处于扁平命名空间中，还是在其自己的另一个层级结构中。

创建虚拟 DIT 视图层次结构的方式与普通的 DIT 层次结构相同。创建相同的条目（例如，组织单元条目），但使用额外的对象类(**nsview**)和描述视图的过滤器属性(***nsviewfilter***)。在添加了附加属性后，与 view 过滤器匹配的条目会立即填充视图。目标条目只 *出现在* 视图中，它们真正的位置不会改变。虚拟 DIT 视图的行为与子树或一级搜索的普通 DIT 的行为可以通过返回预期的结果来执行。

有关添加和修改条目的详情，请参考 *红帽目录服务器管理指南中的*"创建目录条目"

图 4.15. 使用视图组合 DIT

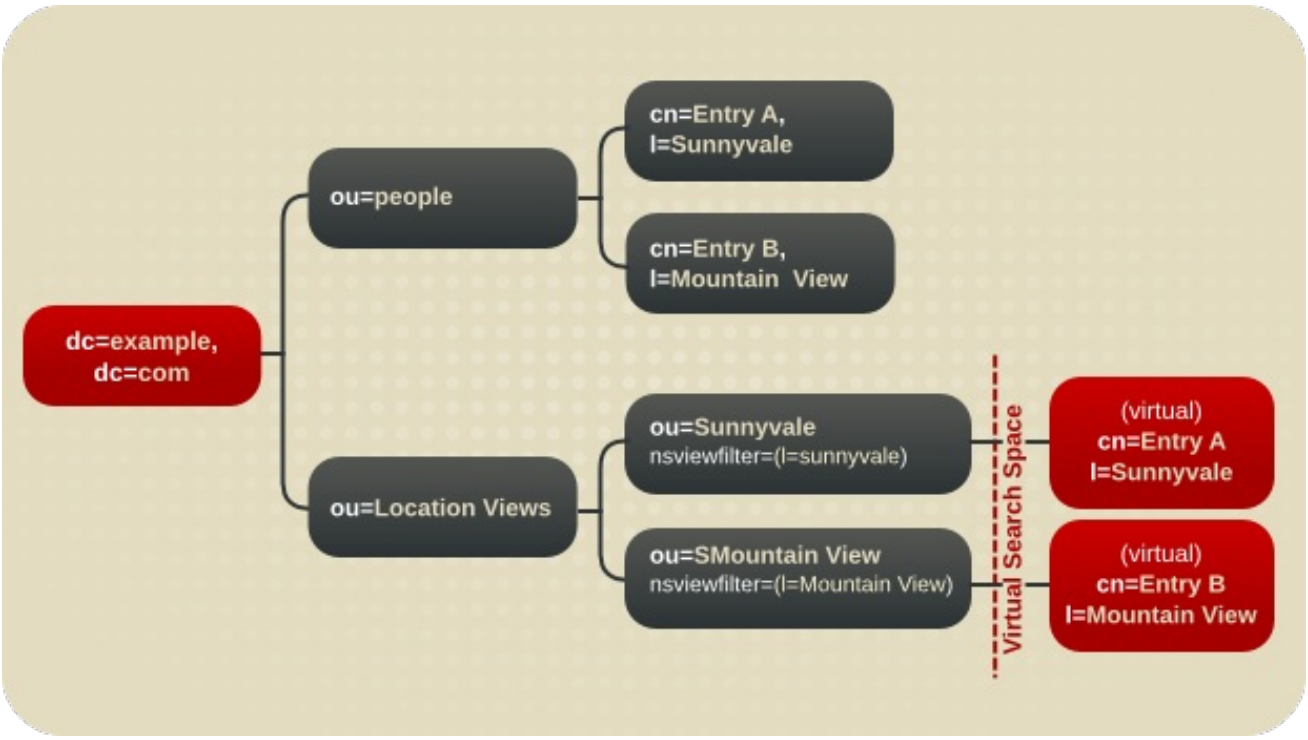


DIT 图 4.15 “使用视图组合 DIT” 演示了当 图 4.14 “Flat 和组织的 DIT 示例” 中显示的两个 DIT 使用视图合并时会发生什么。因为视图本质上允许条目出现在视图层次结构中的多个位置上，因此这个功能已被扩展来根据位置或产品查看销售条目。

假定一组虚拟 DIT 视图层次结构，一个目录用户可以使用视图来导航到所需的条目。例如，如果目标条目是位于 Mountain View 的人员，则首先浏览基于位置的信息最合适。如果是组织性问题，组织视图将是更好的选择。这两个视图同时存在于目录服务器中，并在同一个条目上运行；不同的视图仅在显示其目录结构版本时具有不同的目标。

图 4.15 “使用视图组合 DIT” **中启用了视图的目录中的条目包含在层次结构中最顶层视图的扁平命名空间中。这不是必须的。条目可以保存在其自身的层次结构中。视图相对于某个条目放置的唯一问题是，它必须是视图层次结构的父项的后代。**

图 4.16. 带有虚拟 DIT 视图层次结构的 DIT



- sub-tree ou=People 包含实际条目 A 和 Entry B 条目。
- sub-tree ou=Location Views 是一个视图层次结构。
- leaf nodes ou=Sunnyvale 和 ou=Mountain View 各自包含一个属性 *nsviewfilter*，它描述了视图。

  这些是叶节点，因为它们不包含实际条目。但是，当客户端应用程序搜索这些视图时，它会在 ou=Sunnyvale 下找到 Entry A，并在 ou=Mountain View 下找到条目 B。这个虚拟搜索空间由所有上级视图的 *nsviewfilter* 属性描述。从视图中进行的搜索会返回虚拟搜索空间和实际搜索空间中的条目。这可让视图层次结构作为传统 DIT 工作，或者将传统的 DIT 更改为 view 层次结构。

4.4.2. 使用虚拟 DIT 视图的好处

部署决策通过虚拟 DIT 视图变得更加容易，因为：

- 视图有助于将扁平命名空间用于条目，因为虚拟 DIT 视图提供了类似于传统层次结构提供的导航和管理器性支持。

  另外，每当对 DIT 进行更改时，条目从不需要移动；只有虚拟 DIT 视图层次结构变化。由于这些层次结构不包含实际条目，因此简单且快速修改。

- 在部署计划期间超额，使用虚拟 DIT 视图的灾难性较低。如果在第一个实例中没有正确开发层次结构，可以在不中断服务的情况下轻松、快速地更改。

- 查看层次结构可在几分钟内完全修订，结果会立即实现，从而显著降低目录维护成本。

  对虚拟 DIT 层次结构的更改会立即实现。发生组织更改时，可以快速创建一个新的虚拟 DIT 视图。新的虚拟 DIT 视图可以与旧视图同时存在，从而促进了更逐步更改自身以及使用这些条目的应用程序。因为目录中的一个机构更改不是全权操作，所以可以在一段时间内完成，无需服务中断。

- 通过使用多个虚拟 DIT 视图进行导航和管理，可以更灵活地使用目录服务。

  通过虚拟 DIT 视图提供的功能，组织可以使用旧方法和新方法组织目录数据，而无需在 DIT 的特定点上放置条目。

- 虚拟 DIT 视图层次结构可以作为某种可用的查询来创建，以方便检索常见必要信息。

- 视图在工作实践中提升灵活性并降低目录用户创建复杂搜索过滤器的要求，使用它们原本不需要知道的属性名称和值。

  能够灵活地查看和查询目录信息，让最终用户和应用程序能够更直观地找到他们通过分层导航功能所需的内容。

4.4.3. 虚拟 DIT 视图示例

下面的 LDIF 条目显示基于位置的虚拟 DIT 视图层次结构。驻留在 dc=example,dc=com 之下的任何条目都显示在此视图中，按位置组织。

```
dn: ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Location Views
description: views categorized by location


dn: ou=Sunnyvale,ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Sunnyvale
nsViewFilter: (l=Sunnyvale)
description: views categorized by location


dn: ou=Santa Clara,ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Santa Clara
nsViewFilter: (l=Santa Clara)
description: views categorized by location


dn: ou=Cupertino,ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Cupertino
nsViewFilter: (l=Cupertino)
description: views categorized by location
```

基于 ou=Location Views,dc=example,dc=com 的子树搜索将返回 dc=example,dc=com 下的所有条目，它与过滤器 (l=Sunnyvale), (l=Santa Clara), 或 (l=Cupertino) 匹配。相反，一级搜索不会返回除子视图条目之外的条目，因为所有限定条目都位于三个下级视图。

ou=Location Views,dc=example,dc=com 视图条目本身不包含过滤器。此功能有助于组织分层，无需进一步限制视图中包含的条目。任何视图都可能省略该过滤器。虽然示例过滤器非常简单，但所使用的过滤器可能很复杂。

可能需要限制视图应包含的输入类型。例如，要将此层次结构限制为仅包含人员条目，请将 *nsfilter* 属性添加到 ou=Location Views,dc=example,dc=com，其过滤器值 (objectclass=organizationalperson)。

每个视图具有过滤器限制所有下级视图的内容，而带有过滤器的下级视图也会限制其上级的内容。例如，首先创建顶部视图 ou=Location Views 和上述新过滤器，会创建一个带有 organization 对象类的所有条目的视图。当添加了进一步限制条目的 descendant 视图时，现在在 descendant 视图中出现的条目会从 ancestor 视图中删除。这演示了虚拟 DIT 视图如何模拟传统 DIT 的行为。

虽然虚拟 DIT 视图模拟了传统的 DIT 的行为，但 view 可以执行传统 DIT 不能的操作：条目可能会出现在多个位置上。例如，要将 Entry B 与 Mountain View 和 Sunnyvale（请参阅 图 4.16 “带有虚拟 DIT 视图层次结构的 DIT”）关联，将 Sunnyvale 值添加到 location 属性中，该条目会出现在这两个视图中。

4.4.4. 查看和其他目录功能

目录服务器中的 *class of service* 和 *roles* 支持视图；请参阅 第 4.3 节 “分组目录条目”。当在视图层次结构中添加类服务或角色时，逻辑上和实际包含在视图中的条目都将被视为在范围内。这意味着，可以使用虚拟 DIT 视图来应用角色和服务类，但应用程序的影响在查询扁平命名空间时也可以看到。

有关使用这些功能的详情，请参考 *红帽目录服务器管理指南中的* "高级条目管理"。

使用视图时，可能需要使用稍有不同的方式来访问控制。因为目前对 ACL 的显式支持，在视图父项中创建基于角色的 ACL，并将角色添加到视图层次结构中的相应部分。这样，利用层次结构的 organizational 属性。

如果搜索的基本是视图，并且搜索范围不是基础，则搜索是一个基于视图的搜索。否则，这是传统的搜索。

例如，执行一个基础为 dc=example,dc=com 的搜索不会返回虚拟搜索空间中的任何条目；实际上，不会执行 virtual-search-space 搜索。只有在搜索基础为 ou=Location Views 时才会进行视图处理。这样，视图可确保搜索不会产生这两个位置的条目。（如果它是传统的 DIT，来自两个位置的条目都会被返回。）

4.4.5. 虚拟视图对性能的影响

基于视图的层次结构性能取决于层次结构本身的结构和 DIT 中的条目数量。通常，如果目录服务中启用了虚拟 DIT 视图，则可能会对性能进行边缘更改（在传统 DIT 上有几分百分比的搜索）。如果搜索没有调用视图，则不会影响性能。根据预期的搜索模式和在部署前加载的虚拟 DIT 视图。

如果视图用作机构中通用的导航工具，我们还建议将视图过滤器中使用的属性索引。此外，当视图使用的子过滤器与配置的虚拟列表视图索引匹配时，该索引用于查看评估。

不需要专门调优目录的任何其他部分，以供视图使用。

4.4.6. 与现有应用程序兼容

虚拟 DIT 视图旨在模拟传统 DIT 高度。存在对于大多数应用程序而言，视图的存在应是透明的；它们不应表示它们正在使用视图。除了一些特殊情况外，用户无需了解在目录服务器实例中使用视图的目录；视图看起来和行为像传统的 DIT 一样。

某些类型的应用程序可能会遇到与启用了视图的目录服务相关的问题。例如：

- 使用目标条目的 DN 来导航 DIT 的应用程序。

  这种类型的应用程序会发现，它正在导航出条目物理存在的层次结构，而不是找到该条目的视图层次结构。这样做的原因是，通过更改条目的 DN 以符合视图的层次结构，不会尝试忽略条目的真正位置。这由设计 - 如果条目的真正位置进行解包，例如依赖 DN 来识别唯一条目的应用程序时，许多应用程序都无法正常工作。这种对 DN 的超额导航是客户端应用程序的一个不常见的技术，但却没有人为无法按预期工作的那些客户端。

- 使用 *numSubordinates* 操作属性的应用程序来确定节点下存在多少个条目。

  对于视图中的节点，这目前只有那些存在于实际搜索空间中的条目数，忽略虚拟搜索空间。因此，应用程序可能无法使用搜索来评估视图。

4.5. 目录树设计示例

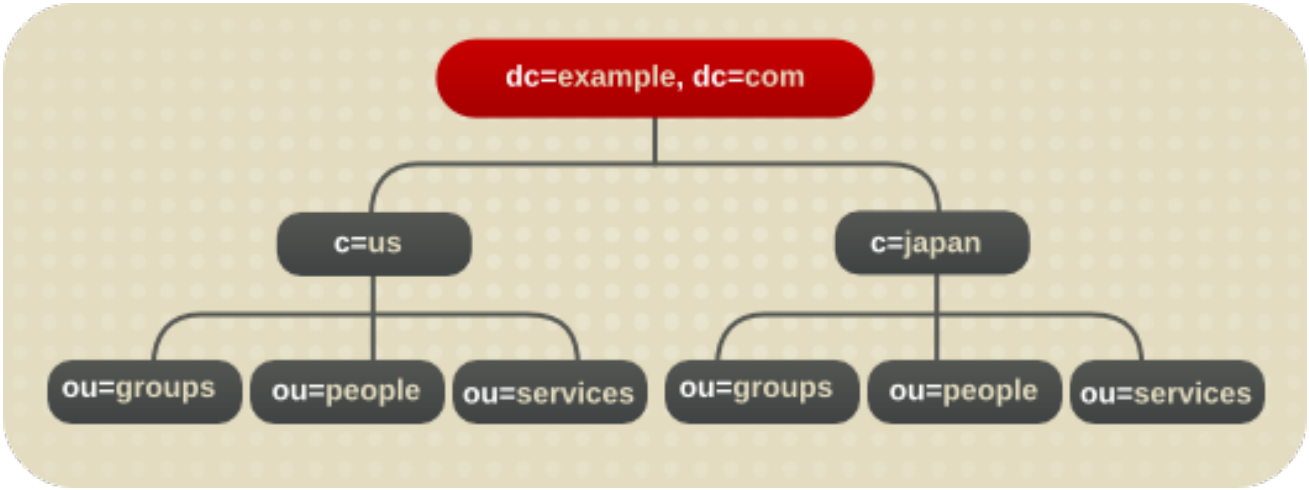
以下小节提供了用于支持扁平层次结构的目录树示例，以及一些更复杂的层次结构示例。

4.5.1. 国际企业的目录树

要支持国际企业，请使用 Internet 域名作为目录树的根点，然后立即为企业有操作的每个国家（位于根点下）进行分支。避免使用国家设计器作为目录树的根点，如 第 4.2.1.1 节 “后缀命名” 所述，特别是当企业是国际的。

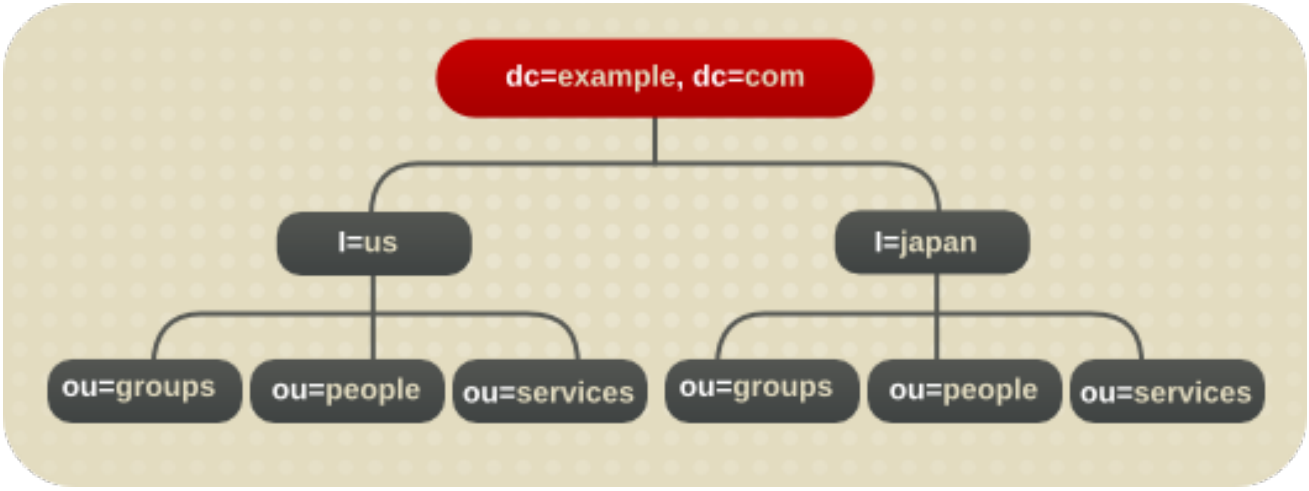
因为 LDAP 对 DN 中属性的顺序没有限制，所以 *c* 属性可以代表每个国家分支：

图 4.17. 使用 c Attribute 来代表不同的计数器



但是，有些管理员认为这是样式的，因此请使用 *l* 属性来代表不同的国家：

图 4.18. 使用 l Attribute 来代表不同的计数器



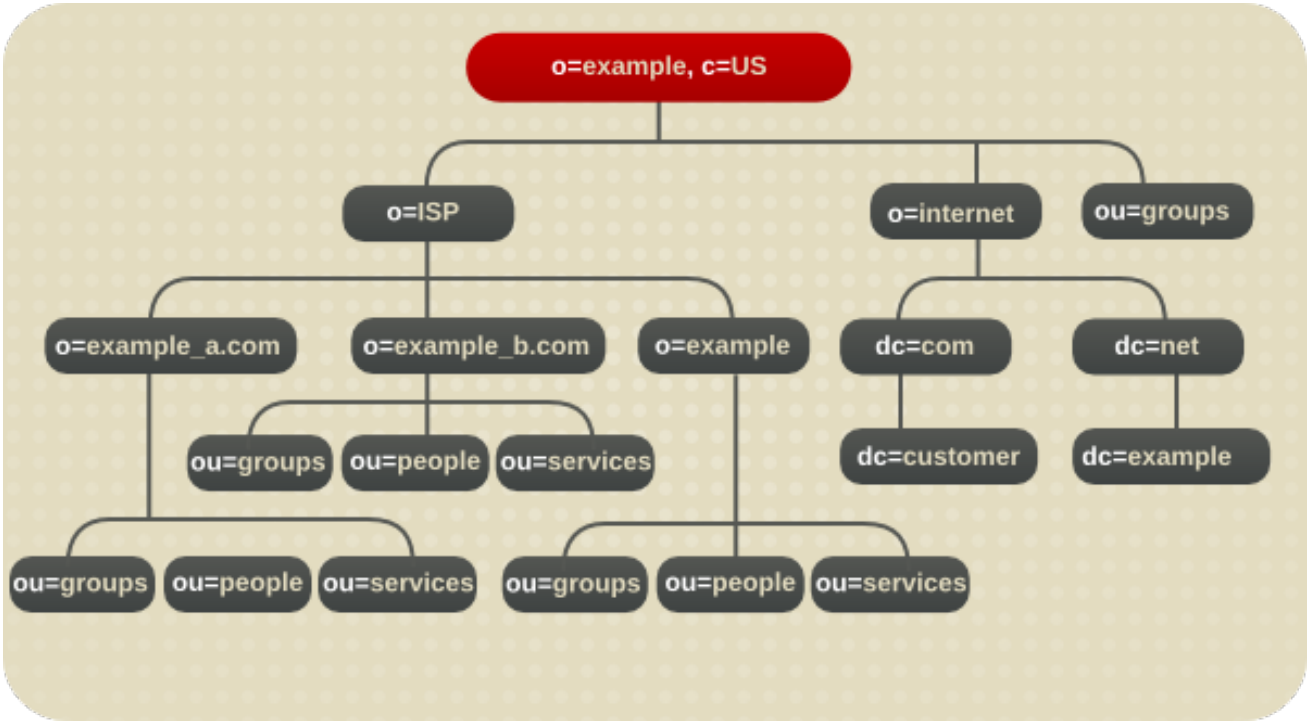
4.5.2. ISP 的目录树

Internet 服务提供商(ISP)可能会支持多个企业及其目录。ISP 应将每个客户视为独特的企业，并相应地设计其目录树。为安全起见，每个帐户应当提供带有唯一后缀和独立安全策略的唯一目录树。

ISP 应该考虑为每个客户分配一个独立数据库，并将这些数据库存储在独立的服务器上。将每个目录树放在自己的数据库中，可以使每个目录树对数据进行备份和恢复，而不影响到其他客户。

另外，分区有助于减少磁盘争用导致的性能问题，并减少受磁盘中断影响的帐户数量。

图 4.19. 示例 ISP 的目录树



4.6. 其他目录树资源

有关设计目录树的更多信息，请参阅以下内容：

- RFC 2247：在 LDAP/X.500 不同的名称中使用域
- RFC 2253 ：LDAPv3、UTF-8 代表的可辨识名称

第 5 章 定义动态属性值

如 第 3.2.2 节 “标准属性” 所述，LDAP 条目将分散部分存储在添加至条目的*属性*中。

Red Hat Directory Server 为动态和自动维护一些目录条目的属性提供了几种不同的机制。这些插件和配置选项简化了管理目录数据并表达条目之间的关系。

5.1. 受管属性简介

在进行站点调查时，最早的步骤之一是识别目录中条目的 *characteristics* (第 2.3.3 节 “为目录数据定性”)。这些特征是需要在目录条目中记录的实体的不同方面。对于员工而言，这意味着个人经理、职务、商业类别、电子邮件地址、家庭和办公室电话号码等信息。条目的每个特性在条目属性中维护。

条目特性的特性是它们相互的*关系*。显然，经理有一个员工，因此与这两个条目相关。组与其成员相关联。不太明显的关系，与共享一个共同物理位置的条目之间也不太明显。

Red Hat Directory Server 提供了几种不同的方法，这些条目之间的关系可以平稳保持且一致。有几个插件可作为目录中的数据的一部分自动应用或生成属性：

- *属性唯一*要求子树或数据库中的特定属性的每个实例都有唯一的值。每当创建条目或修改属性时，都会强制实施。
- *服务类*将一个条目用作模板；每当该属性值更改时，CoS 范围内的所有其他条目都会在其条目上自动具有相同的属性。（受 CoS 影响的条目通过定义条目来标识。）
- 当创建定义的范围的另一个条目时，*受管条目*会根据定义的模板创建一个条目。有时，特别是与外部客户端集成时，可能需要自动创建和管理条目对。受管条目定义第二个条目的模板，并提供自动更新的机制。
- *链接的属性*遵循一个条目中的属性中的 DN 值，并将预先确定的属性（具有指向原始条目的值）到引用的条目。因此，如果条目 A 列出条目 B 作为直接报告，则可以自动更新条目 B，使其包含 *manager* 属性，并将条目 A 作为指定的管理器。
- *分布式数字分配*会自动为条目分配唯一标识号。这对 GID 或 UID 号分配有用，这在组织内必须是唯一的。

考虑有关特定条目属性值的几个方面，作为规划目录数据和目录模式的一部分：

- 条目是如何相关？条目之间有哪些常见属性？是否存在哪些属性必须代表条目间的连接？
- 数据的原始源可能被维护，在哪里以及位置（在什么条目中）？此信息更新的频率以及数据更改时影响多少条目？
- 这些条目使用哪些架构元素以及这些属性的语法是什么？
- 插件如何处理分布式目录配置，如复制或同步？

5.2. 关于属性唯一性

Attribute Uniqueness 插件是一个预先合作插件。这意味着，插件会在服务器执行 LDAP 操作 *前检查* 所有更新操作。该插件决定操作是否应用到某一属性以及它配置为监控的后缀。

如果更新操作应用到插件监控的属性和后缀，并导致两个条目具有相同的属性值，则服务器终止该操作并将 LDAP_CONSTRAINT_VIOLATION 错误返回给客户端。

属性插件的每个实例都在一个或多个子树的单个属性上执行检查。要检查多个属性的唯一性，必须为每个要检查的每个属性创建一个插件实例。

Attribute Uniqueness 插件可以采用特定的用户定义的方法操作：

- 它可以检查指定子树中的每个条目。

  例如，如果某个公司是 example.com，则托管 example_a.com 和 example_b.com 的目录，当一个条目（如 uid=jdoe,ou=people,o=example_a,dc=example,dc=com ）中被添加，唯一性需要只在 o=example_a,dc=example,dc=com 子树中强制执行。这可以通过在属性唯一标识符配置中明确列出子树的 DN。

- 指定与更新条目 DN 中的条目相关的对象类，并对它下的所有条目执行唯一性检查。

  这个选项在托管环境中很有用。例如，当添加诸如 uid=jdoe,ou=people, o=example_a,dc=example,dc=com 等条目时，在 o=example_a,dc=example,dc=example,dc=com 子树中显式列出此子树时，通过指示 *标记对象类*。如果标记对象类设置为 机构，则唯一性检查算法会在 DN 中查找具有此对象类 (o=example_a)的条目，并对它下的所有条目执行检查。

  另外，只有在更新的条目包含指定对象类时，才能检查唯一性。例如，只有在更新的条目包含 objectclass=inetorgperson 时，才能执行检查。

当目录服务器首次设置时，目录服务器为 ***uid*** 属性提供属性唯一插件的默认实例。此插件实例确保提供给 ***uid*** 属性的值在 root 后缀中是唯一的（与 userRoot 数据库对应的后缀）。

此插件默认是禁用的，因为它会影响多层次复制的操作。

当更新作为复制操作的一部分执行时，属性插件不会对属性值执行任何检查。

由于客户端应用程序的所有修改都在供应商服务器上执行，因此应在供应商上启用属性插件。在消费者服务器上启用它是不需要的。

在消费者上启用属性唯一插件不会阻止目录服务器正确运行，但可能导致性能下降。

在多层次复制方案中，供应商同时充当同一副本的供应商和消费者。由于多组复制使用松散一致的复制模型，因此在一台服务器上启用属性唯一性插件不足，从而确保在任意给定时间在两个供应商服务器中都是唯一的属性值。因此，在一个服务器上启用属性插件可能会导致每个副本中保存的数据不一致。

但是，可以使用属性唯一插件，从而满足以下条件：

- 执行唯一性检查的属性是 naming 属性。
- 在这两个供应商服务器上都启用了属性唯一插件。

满足这些要求时，复制时会报告属性唯一性冲突。命名冲突需要手动解析。

5.3. 关于服务类

服务类服务 (CoS)在对应用程序不可见的条目间共享属性。使用 CoS 时，可能不能使用条目本身存储一些属性值。相反，它们将由服务逻辑类生成，因为该条目将发送到客户端应用程序。

例如，该目录包含数千个条目，它们共享了通用属性 ***facsimileTelephoneNumber***。传统上，修改传真号需要单独更新每个项，这可能会是一个非常大型的任务，并可能造成没有全部更改的风险。使用 CoS 时，可以动态生成属性值。***facsimileTelephoneNumber*** 属性存储在一个位置，每个条目都从该位置检索其 fax number 属性。对于应用程序，这些属性看起来像所有其他属性一样，尽管实际存储在条目本身上。

每个 CoS 都由目录中的几个条目组成：

- *CoS 定义条目*标识 CoS 的类型。它作为 LDAP 子条目存储在它所影响的分支下的 LDAP 子条目中。
- *template 条目*包含 shared 属性值的列表。对模板条目属性值的更改会自动应用到共享属性的所有条目。

CoS 定义条目和模板条目交互为*目标条目*（其范围内的条目）提供属性值。它们提供的值取决于以下内容：

- 条目的 DN（目录树中的不同部分可能包含不同的 CoS）。
- 通过该条目存储的服务类属性值。

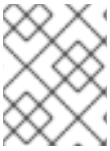
  缺少 service class 属性可以表示特定的默认 CoS。
- 存储在 CoS 模板条目中的属性值。

  每个 CoS 模板条目提供特定 CoS 的属性值。
- 条目的对象类。

  只有条目包含对象类时，才会生成 COS 属性值，允许启用架构检查时属性；否则，将生成所有属性值。
- 存储在目录树中某个特定条目的属性。

重要

不要索引您在 CoS 定义中使用的属性(***cosAttribute*** 参数)。



注意

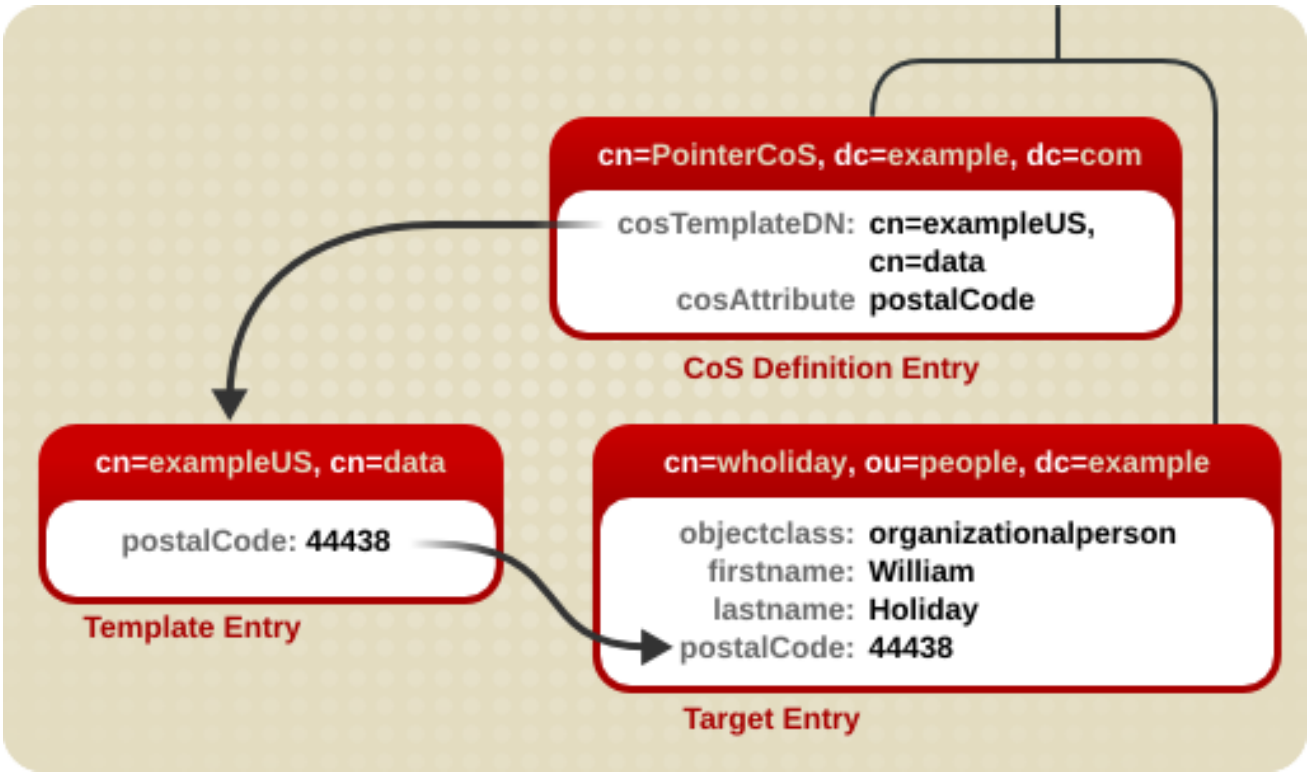
角色和经典 CoS 一起提供基于角色的属性。这些属性会出现在条目上，因为它具有关联 CoS 模板的特定角色。例如，使用基于角色的属性来基于角色设置服务器查找限制。

5.3.1. 关于 Pointer CoS

指针 CoS 仅使用模板 DN 识别模板条目。每个指针的 CoS 只能有一个模板 DN。pointer CoS 适用于模板条目范围内的所有条目。

例如，图 5.1 “Pointer CoS 示例” 中的指针 CoS 共享一个通用邮政代码，其中包含存储在 **dc=example,dc=com** 下的所有条目。

图 5.1. Pointer CoS 示例

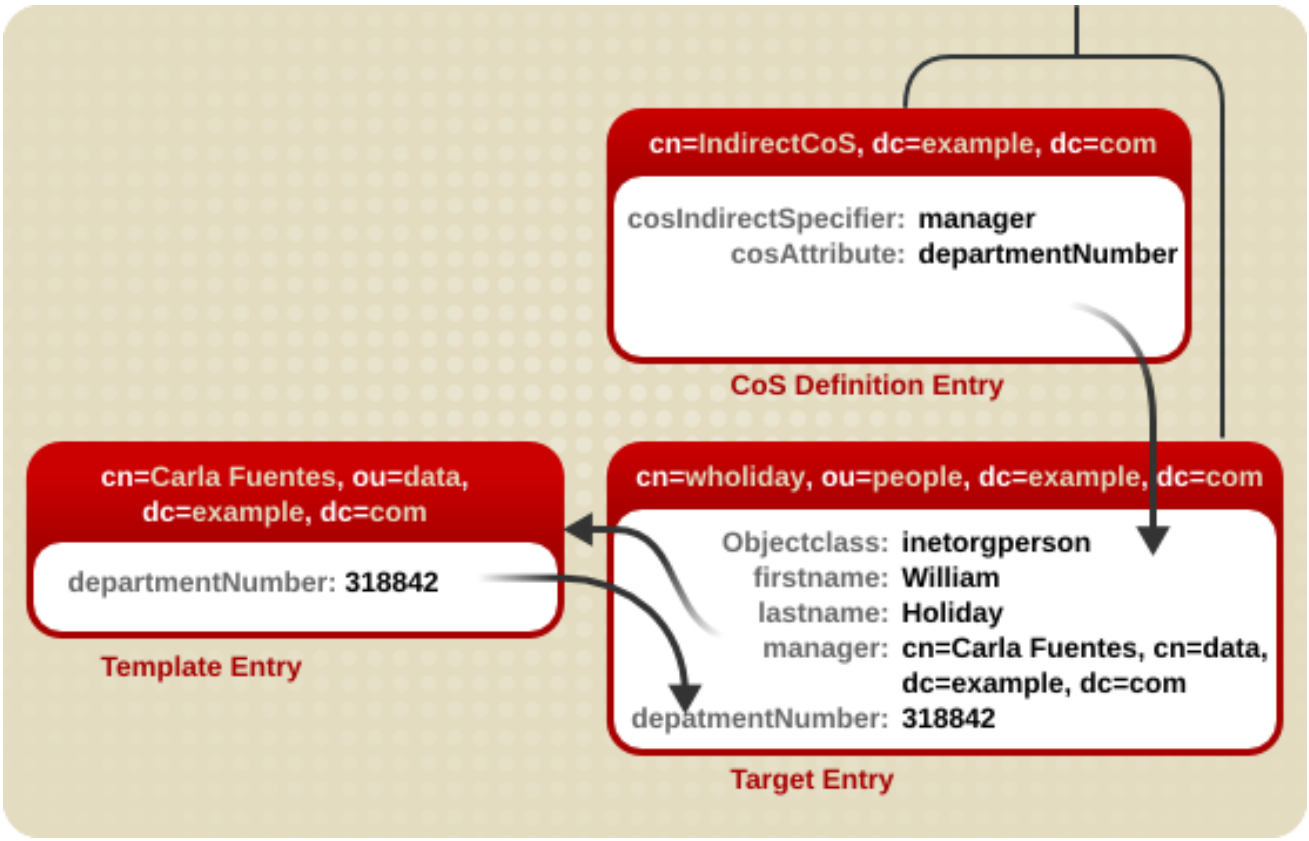


模板条目由 CoS 定义条目的 DN cn=exampleUS,cn=data 标识。每次在条目 cn=wholiday,ou=people,dc=example,dc=com 上查询 *postalCode* 属性时，Directory 服务器都会返回模板条目 cn=exampleUS,cn=data 中可用值。

5.3.2. 关于Indirect CoS

间接 CoS 使用目标条目属性之一的值来识别模板条目。target 条目的属性必须包含现有条目的 DN。

图 5.2. Indirect CoS 示例

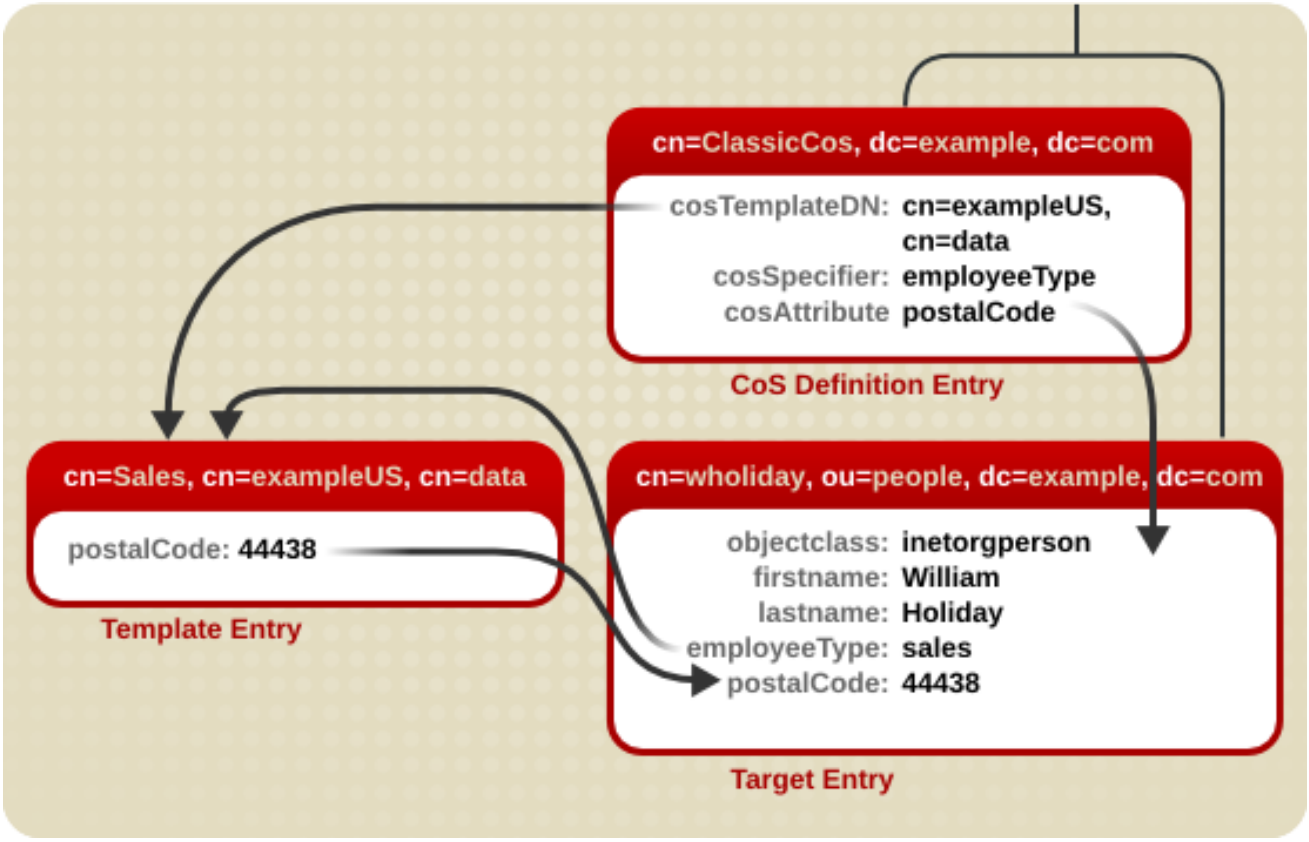


在本例中，William Holiday 的目标条目包含 *manager* 属性的间接指定符。William 的经理是 Carla Fuentes，因此 *manager* 属性包含模板条目的 DN 的指针 cn=Carla Fuentes,ou=people,dc=example,dc=com。模板条目依次提供 *departmentNumber* 属性值 318842。

5.3.3. 关于经典 CoS

典型的 CoS 通过其 DN 和其中一个目标条目的属性来识别*模板条目*。经典的 CoS 可以有多个模板条目，包括要应用到不属于任何其他 CoS 模板的条目。

图 5.3. Classic CoS 示例



在本例中，CoS 定义条目的 ***cosSpecifier*** 属性指定 ***employeeType*** 属性。此属性与模板 DN 相结合，将模板条目标识为 **cn=sales,cn=exampleUS,cn=data**。然后，模板条目为目标条目提供 ***postalCode*** 属性的值。

5.4. 关于受管条目

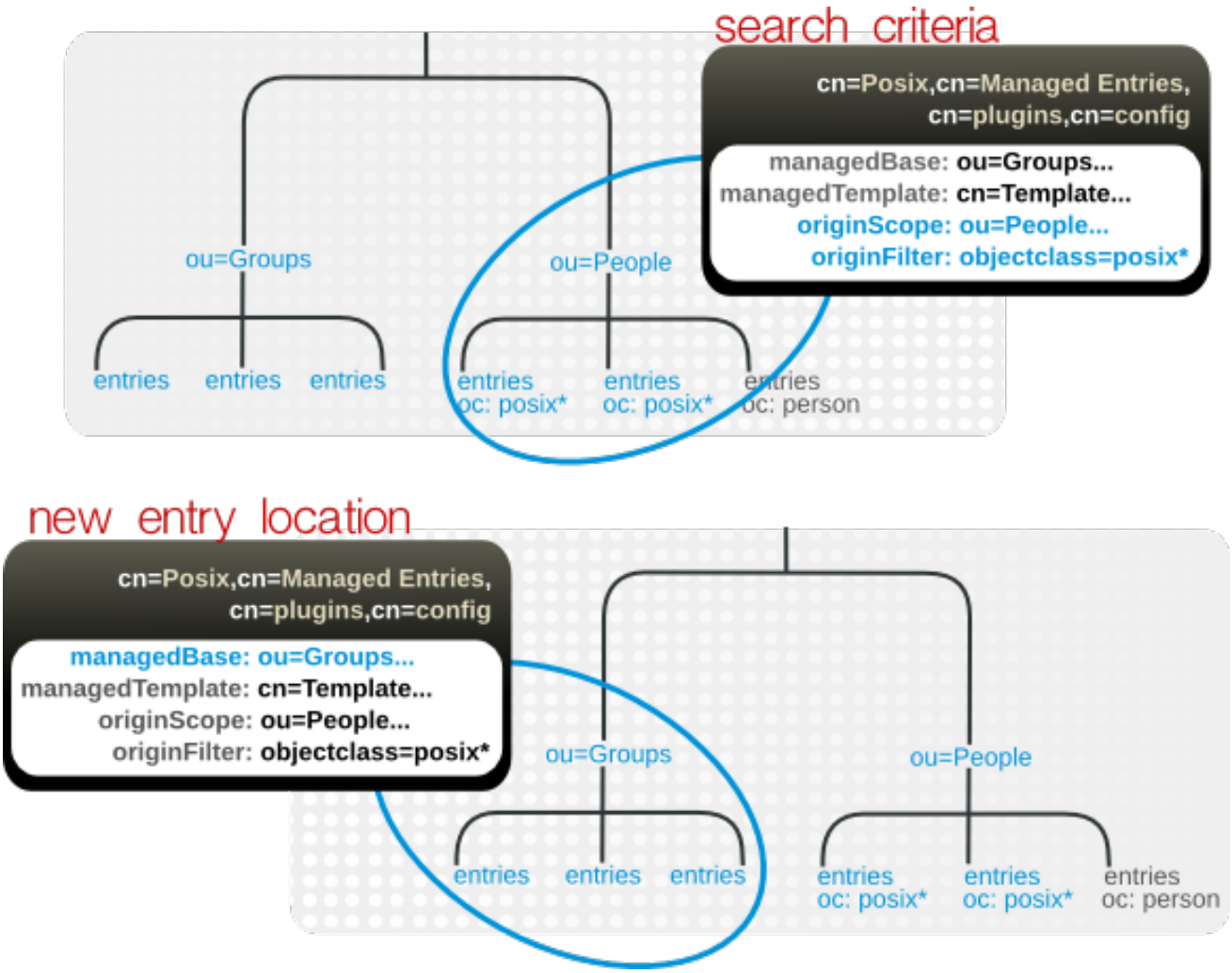
有些客户端与 Red Hat Directory Server 集成需要双条目。例如，Posix 系统通常为每个用户有一个组。Directory 服务器的 Managed Entries 插件会在创建适当的原始卷条目时自动创建一个新的受管条目，其属性准确和特定值会自动进行。

基本的概念是，在创建 Entry A 时，应该自动使用一个带有相关属性值的 Entry B。例如，当创建 Posix 用户(**posixAccount** 条目)时，还应创建对应的组条目(**posixGroup** 条目)。Managed Entries 插件的实例标识哪个条目（*原始条目*）会触发插件自动生成新条目（*受管条目*）。它还标识了定义受管条目的单独模板条目。

Managed Entries 插件的实例定义了三个内容：

- 用于标识原始条目的搜索条件（使用搜索范围和搜索过滤器）
- 在其中创建受管条目（新条目位置）的子树
- 用于受管条目的模板条目

图 5.4. 定义受管条目



例如：

```
dn: cn=Posix User-Group,cn=Managed Entries,cn=plugins,cn=config
objectclass: extensibleObject
cn: Posix User-Group
originScope: ou=people,dc=example,dc=com
originFilter: objectclass=posixAccount
managedBase: ou=groups,dc=example,dc=com
managedTemplate: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
```

原始条目不必具有创建受管条目的任何特殊配置或设置；只需在插件范围内创建它，并与给定的搜索过滤器匹配。

5.4.1. 为受管条目定义模板

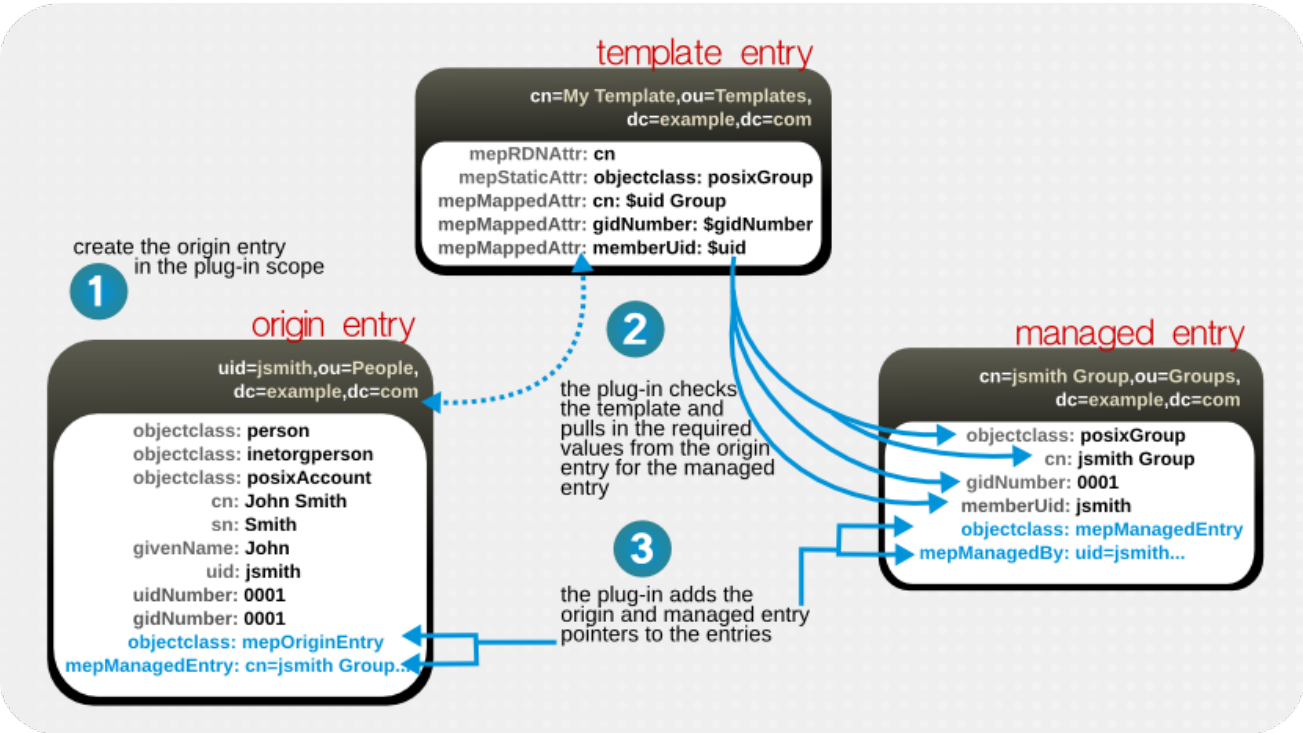
模板条目使用静态属性（带有预定义值的一个和映射属性）和映射的属性（从原始条目中提取其值）的整个配置。

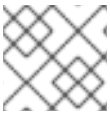
```
dn: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
objectclass: mepTemplateEntry
cn: Posix User-Group Template
mepRDNAttr: cn
mepStaticAttr: objectclass: posixGroup
```

```
mepMappedAttr: cn: $uid Group
mepMappedAttr: gidNumber: $gidNumber
mepMappedAttr: memberUid: $uid
```

模板中映射的属性使用令牌（以美元符号($)开头），以从 origin 条目中提取值并在受管条目中使用它。

图 5.5. 受管条目、模板和原始条目





注意

确保为静态和映射的属性赋予的值符合所需的属性语法。

5.4.2. Managed Entries 插件的 entry Attributes Written

原始条目和受管条目都具有特殊的受管条目属性，表明它们由 Managed Entries 插件的实例管理。对于原始条目，插件会添加指向关联的受管条目的链接。

```
dn: uid=jsmith,ou=people,dc=example,dc=com
objectclass: mepOriginEntry
objectclass: posixAccount
...
sn: Smith
mail: jsmith@example.com
mepManagedEntry: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
```

除了模板中定义的属性外，插件还会添加指向原始条目的属性。

```
dn: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
objectclass: mepManagedEntry
objectclass: posixGroup
...
mepManagedBy: uid=jsmith,ou=people,dc=example,dc=com
```

使用特殊属性来指示受管和原始条目，可以轻松地识别相关条目并评估由 Managed Entries 插件所做的更改。

5.4.3. Managed Entries 插件和目录服务器操作

Managed Entries 插件对目录服务器执行常见操作（如添加和删除操作）有一些影响：

- *添加*。对于每个添加操作，服务器会检查新条目是否在任何 Managed Entries 插件实例范围内。如果满足原始条目的条件，则创建受管条目和受管条目相关的属性将添加到 origin 和 managed 条目中。

- *修改*。如果修改了原始条目，它会触发插件来更新受管条目。

  但是，更改 *模板* 条目不会自动更新受管条目。对模板条目的任何更改都不会反映在受管条目中，直到下次修改原始条目后。

  在受管条目 *中*映射的受管属性无法手动修改，只有通过 Managed Entry 插件进行修改。受管条目中的其他属性（包括由 Managed Entry 插件添加的静态属性）可以手动修改。

- *删除*。如果删除了 origin 条目，则 Managed Entries 插件也会删除与该条目关联的任何受管条目。

  对可以删除的条目有一些限制。

  - 如果模板条目当前由插件实例定义引用，则无法删除它。

  - 除了 Managed Entries 插件外，无法删除受管条目。

- *重命名*。如果重命名了原始条目，则插件会更新对应的受管条目。如果条目从插件范围 *移出*，则删除受管条目；而如果某个条目 *移至* 插件范围，它将被视为 add 操作，并且创建新的受管条目。

  与删除操作一样，可以重命名或移动条目受到限制。

  - 无法将配置定义条目从 Managed Entries 插件容器条目中移出。如果删除了该条目，则该插件实例将处于激活状态。

  - 如果条目 *移至* Managed Entries 插件容器条目，则它将被验证并视为活跃的配置定义。

  - 如果模板条目目前由插件实例定义引用，则无法重命名或移动。

  - 除了 Managed Entries 插件外，无法重命名或移动受管条目。

- *复制。复制更新不会启动* Managed Entries 插件操作。如果插件范围中某个条目的添加或修改操作被复制到另一个副本，则该操作不会触发副本上的 Managed Entries 插件实例来创建或更新条目。要复制受管条目的更新的唯一方法是将最终受管条目复制到副本。

5.5. 关于链接属性

服务类动态提供条目的属性值，*它们具有相同的值* 的属性，如构建地址、后代代码或主要办公室号码。这些是共享属性值，它在单个模板条目中更新。

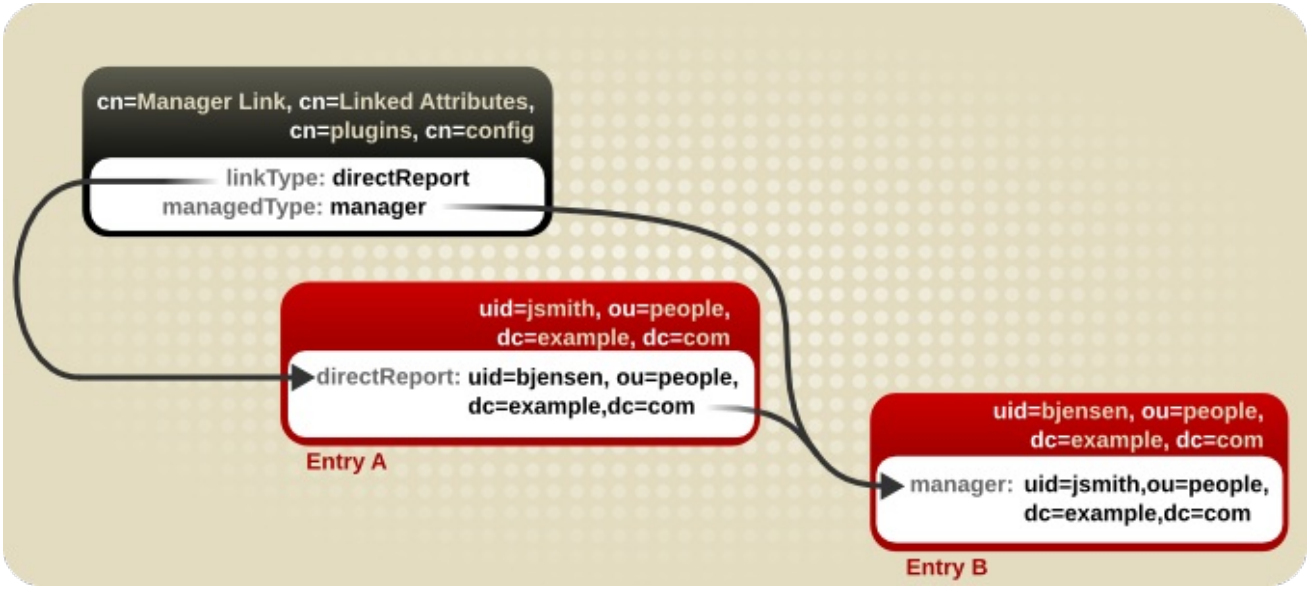
但是，通常情况下，不同条目之间需要有一种表达其之间的链接方式，但显示该关系的值（甚至可能是属性）。Red Hat Directory Server 提供了一种将指定属性链接在一起的方法，以便在一个条目中的一个属性被改变时，相关条目上的相应属性会被自动更新。第一个属性具有一个指向要更新的条目的 DN 值；第二个条目属性也具有 DN 值，它是第一个条目的 back-pointer。

例如，组条目在诸如 *成员* 的属性中列出其成员。在用户所属的组的用户条目中，可以指示其中的一个问

题；这在 ***memberOf*** 属性中设置。***memberOf*** 属性是通过 MemberOf 插件的*managed* 属性。该插件轮询每个组条目，以针对其各自成员属性的更改。每当从组中添加或删除组成员时，对应的用户条目都会使用更改的 ***memberOf*** 属性进行更新。这样，*成员*（和其他成员属性）和 ***memberOf*** 属性 *链接*。

MemberOf 插件仅限于单个实例，仅适用于一个（单一）组成员属性（一些其他行为对组（如处理嵌套组）是唯一的。另一个插件是 Linked Attributes 插件，允许多个插件的实例。每个实例配置一个属性，它由管理员手动维护(***linkType***)和一个属性，由插件(***managedType***)自动维护。

图 5.6. 基本链接属性配置

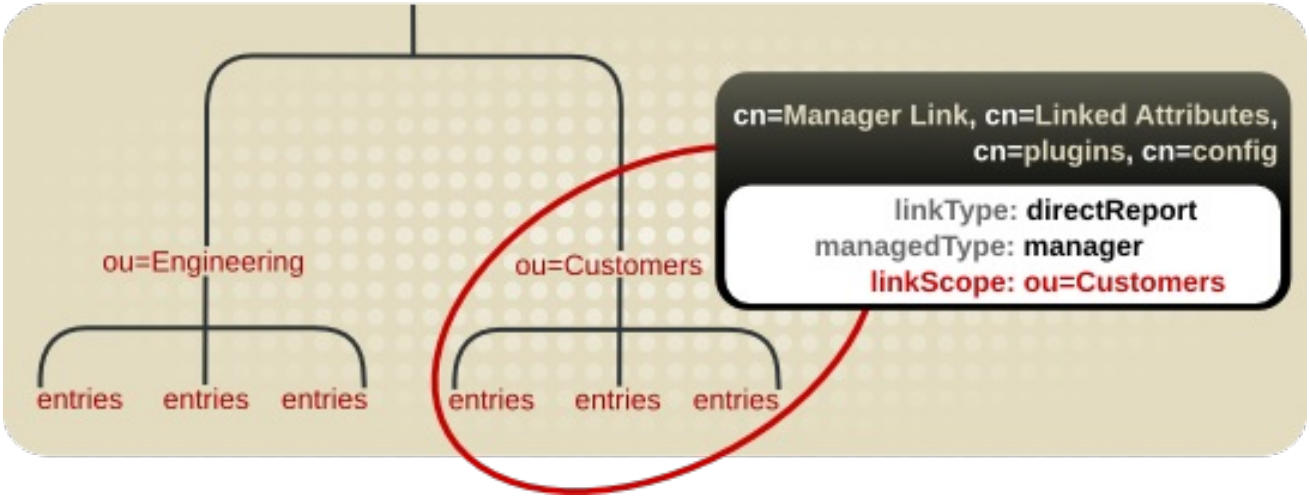


注意

为了保持数据一致性，只有插件进程才会维护 managed 属性。考虑创建一个将限制任何受管属性的所有写入访问权限的 ACI。

一个 Linked Attribute Plug-in 实例可以限制为目录中的单个子树。这可允许对属性组合和受影响的条目进行更加灵活的自定义。如果没有设置范围，则插件对整个目录执行操作。

图 5.7. 限制链接的属性插件到特定子树

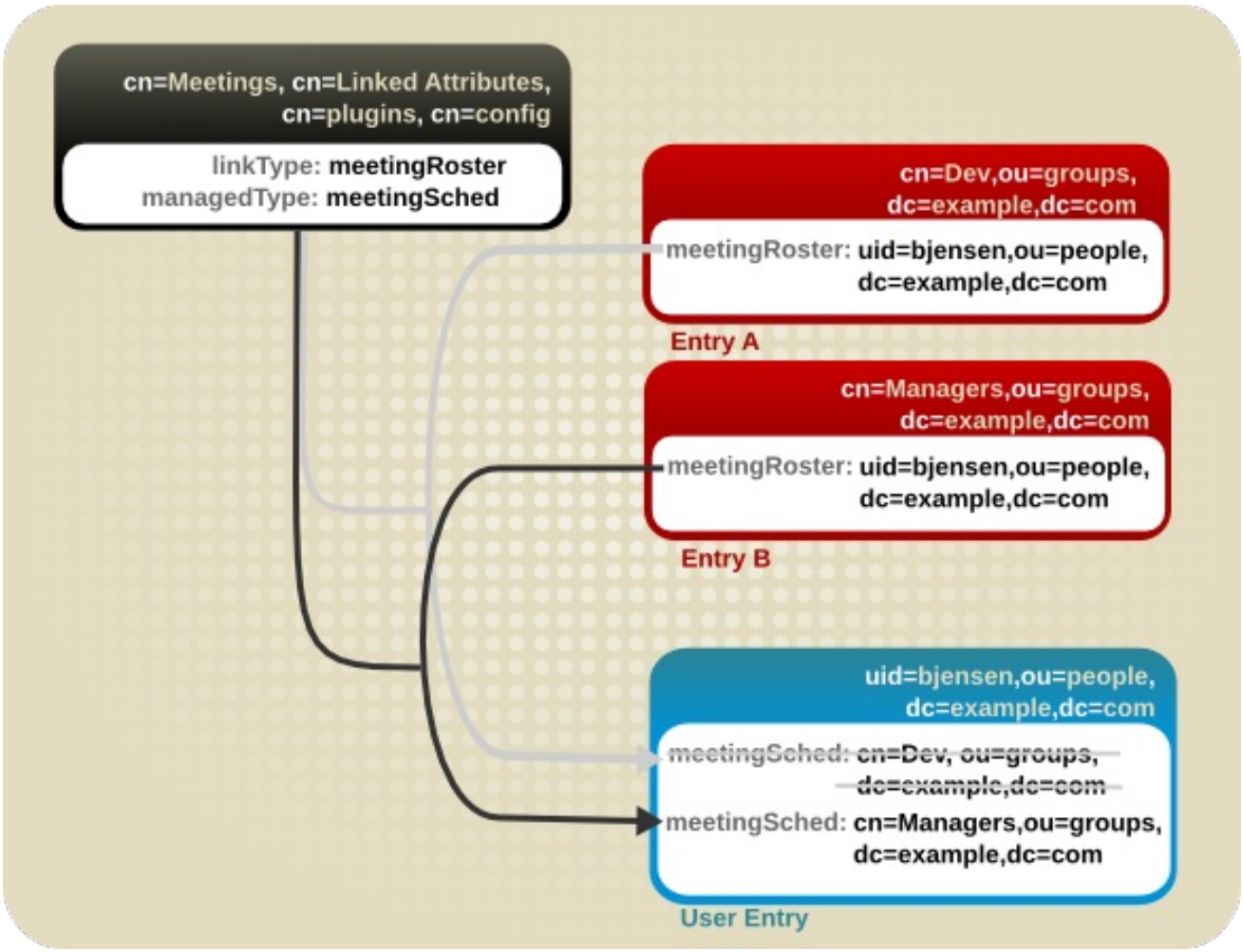


5.5.1. 链接属性的 schema 要求

managed 属性和链接的属性都必须在其属性定义中需要 Distinguished Name 语法。该插件通过从 link 属性拉取 DN 来识别要维护的条目，然后它会自动将原始条目 DN 分配为受管属性值。这意味着这两个属性都必须将 DN 用作值。

managed 属性必须是多值。用户可以是多个组的成员，可以是多个文档的作者，或者具有多个"查看"参考条目。如果 managed 属性是单值，则值不会被正确更新。因为很多标准元素是多值，所以这个问题并不是默认 schema 的大部分问题。但是，在使用自定义 schema 时，特别需要考虑。

图 5.8. 错误：使用单值链接的属性

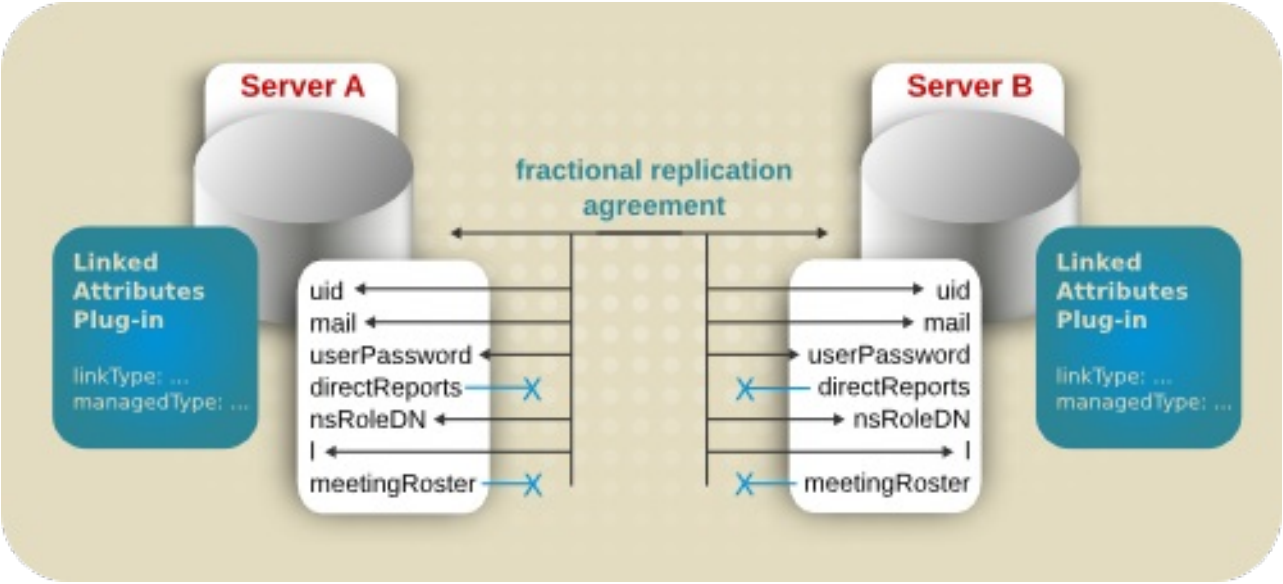


5.5.2. 使用带有复制的链接属性

在简单的复制方案中(supplier-consumer)，然后插件必须仅存在于供应商上，因为没有写入用户才可以在消费者中进行。

对于多supplier 复制，每个供应商必须具有自己的插件实例、所有配置相同的插件实例，并且管理的属性必须使用部分复制来排除在复制中。

图 5.9. 链接属性和复制



对一个供应商所做的任何更改都会自动触发插件，以管理对应目录条目上的值，因此数据可以在服务器间保持一致。但是，受管属性必须由插件实例维护，才能在链接条目之间保持一致。这意味着，受管属性值应完全由插件进程维护，而不是复制过程，即使在多层次复制环境中也是如此。

5.6. 关于动态分配唯一数量值

有些条目属性需要具有唯一数字，如 *uidNumber* 和 *gidNumber*。目录服务器可使用分布式 Numeric Assignment(DNA)插件为指定属性自动生成和提供唯一数字。

很多情况需要一个唯一的数字属性，如 UID/GID 号或 PIN 编号。服务器使用 DNA 插件实例指定生成数字的属性，因此每当将该属性添加到条目时，都可分配唯一的值。

注意

在 DNA 插件中，*属性一致性*不一定会保留。该插件只分配非重叠范围，但它允许为受管属性手动分配数字，而且不会验证或要求手动分配的数字是唯一的。

5.6.1. 目录服务器管理唯一数字的方式

分配唯一数字的问题没有生成数字，但实际上管理数字，在复制条目时不会与其他分配数字冲突，并且每个服务器都有足够数量的分配数字。

服务器的 DNA 插件分配实例可以发出的一系列可用数字。范围定义非常简单，由两个属性设置：服务器的下一个可用数字（范围较低）及其最大值（范围最顶层）。在配置了插件实例时，会设置初始底部范围。之后，底部值由插件更新。通过将可用数字拆分为范围，服务器可以持续分配数字，而不会相互重叠。

服务器在内部执行排序搜索，以查看是否已获取下一个指定范围，这要求 managed 属性具有等同顺序匹配规则的索引。

对于多层次复制，每个供应商都可以配置阈值，以便在其范围内从数字开始运行，可以请求来自其他供应商的其他范围。每个供应商都在单独的配置条目中保持其当前范围的跟踪。配置条目将复制到所有其他供应商，因此每个供应商都可以检查配置以查找要联系新范围的服务器。

在各个服务器和范围配置条目上设置的范围是目录服务器高效*分配*数字的项。

DNA 插件可以分配唯一数字到单个属性类型，或者从单个范围唯一数字的多个属性类型分配。

这提供了为属性分配唯一数字的多个选项：

- 从单一唯一数字范围内分配给单个属性类型的单个数字。
- 对于一个条目，分配给两个属性的唯一数字相同。
- 分配了两个不同的属性，与相同范围的唯一数字不同。

在很多情况下，为每个属性类型分配唯一的数字就足够了。为新的员工条目分配 ***employeeID*** 时，务必要为每个员工条目分配一个唯一的 ***employeeID***。

然而，在有些情况下，从相同数量分配唯一数字到多个属性可能很有用。例如，当将 ***uidNumber*** 和 ***gidNumber*** 分配给 **posixAccount** 条目时，可将 DNA 插件配置为为这两个属性分配相同的数字。

DNA 插件将始终应用于目录树的特定区域（*scope*）以及该子树中的特定条目类型（ *filter*）。

通常，完全不同的用户存储在目录树的不同分支中。例如，托管服务可能在 **ou=Example Corp.** 分支中有一个客户端的用户，在 **ou=Acme Company** 分支中另一个客户端的用户。在这种情况下，分配的数字必须在 ***子树中唯一，*** 但不一定在整个目录中是唯一的。在这种情况下，**ou=Example Corp.** 分支中的 Barbara Jensen 都正确，在她的条目中有 **uidNumber:5**，对于 **ou=Acme Company** 分支中的 John Smith，在其条目中有 **uidNumber:5**，因为这些是单独的机构。***将范围应用到特定的子树在 DNA 范围内设置，如 dnaScope: ou=people,dc=example,dc=com。***

唯一数字也可以通过使用前缀来识别不同类型的用户条目来区分不同的范围。例如，如果将 DNA 前缀设置为 acme，则 Acme Company 分支中的唯一数字在数字前面有 acme，如 uid: acme5。

5.6.2. 使用 DNA 分配值到属性

Directory 服务器可以处理生成属性值的不同方法。

在最简单的情形中，用户条目会添加到具有对象类的目录中，它要求 unique-number 属性，但没有属性。在不使用值的情况下添加受管属性（或需要）将触发 DNA 插件来分配值。添加条目时，该插件会根据插件的范围和过滤器，检查条目是否与定义的范围匹配。如果条目与范围匹配，并且添加条目中缺少该范围管理的属性，那么 DNA 插件将分配下一个值。只有在已经配置了 DNA 插件来为单个属性分配唯一值时，这个选项才起作用。

例如，posixAccount 对象类需要 uidNumber 属性。如果 uidNumber 属性由 DNA 插件管理，并且添加用户条目时没有过滤器范围内的 uidNumber 属性，那么服务器会检查新条目，查看它需要 managed uidNumber 属性，并使用自动分配的值添加属性。

```
ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: posixAccount
uid: jsmith
cn: John Smith
....
```

该插件处理缺少的属性，从服务器请求下一个可用数量，并提供该条目的值。

相似且更可管理的选项是使用一个 魔法号。这个音量号是 managed 属性的模板值、服务器范围之外的内容、数字甚至一个单词，插件可识别它需要替换为新分配的值。当使用该数字添加条目时，该条目位于配置的 DNA 插件的范围和过滤中，然后使用 magic number 自动触发插件来生成新值。

当 DNA 插件被配置为为 uidNumber 和 gidNumber 都分配相同的唯一数字到 posixAccount 条目，DNA 插件将为这两个属性分配相同的数字。为此，请将两个受管属性传递给修改操作，指定数量。例如：

```
ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: posixAccount
uid: jsmith
cn: John Smith
uidNumber: magic
gidNumber: magic
....
```

Magic number 对于从 LDIF 导入条目或触发 DNA 插件来为多个不同属性生成唯一数字，这个数字非常有用。

DNA 插件仅生成新的、唯一的值。如果向条目添加或修改为 DNA 插件控制的属性使用特定值，则会使用指定的数字；DNA 插件不会覆盖它。

注意

在 DNA 插件中，属性一致性不一定会保留。该插件只分配非重叠范围，但它允许为受管属性手动分配数字，而且不会验证或要求手动分配的数字是唯一的。

5.6.3. 使用带有复制的 DNA 插件

使用多层次复制时，服务器会引用两个条目：

- DNA 插件的受管范围
- 保存服务器可用范围信息的共享配置条目

创建插件实例时，DNA 插件会自动在带有供应商配置的共享配置条目下创建一个条目。例如：

```
dn: dnaHostname=ldap1.example.com+dnaPortNum=389,cn=Account
UIDs,ou=Ranges,dc=example,dc=com
objectClass: extensibleObject
objectClass: top
dnahostname: ldap1.example.com
dnaPortNum: 389
dnaSecurePortNum: 636
dnaRemainingValues: 1000
```

当服务器需要新的数量时，它会搜索容器条目下的配置条目。当服务器找到可用范围最高的服务器时，它会发送扩展操作请求，使其具有为其分配的范围的一部分。如果第二个服务器同意，第二服务器会向请求服务器发送新范围分配。

第 6 章 设计目录拓扑

第 4 章 设计目录树 涵盖目录服务存储条目的方式。由于红帽目录服务器可以存储大量条目，因此可以在多个服务器间分发目录条目。目录的拓扑描述了如何将目录树划分为多个物理目录服务器以及这些服务器如何相互链接。

本章论述了规划目录服务的拓扑。

6.1. 拓扑概述

目录服务器可以支持 分布式目录，其中目录树（在 第 4 章 设计目录树 中指定）分布到多个物理目录服务器中。目录划分到这些服务器的方法有助于实现以下内容：

- 获得启用目录的应用程序的最佳性能。
- 提高目录服务的可用性。
- 改进目录服务的管理。

数据库是作业的基本单元，如复制、执行备份和恢复数据。单个目录可以划分为可管理片段，并分配到单独的数据库。然后，这些数据库可以在多个服务器间分布，从而减少每台服务器的工作负载。多台数据库可以位于单一服务器上。例如，一个服务器可能包含三个不同的数据库。

当目录树划分为多个数据库时，每个数据库都包含目录树的一个部分，称为后缀 (suffix)。例如，一个数据库可用于仅存储目录树的 ou=people,dc=example,dc=com 后缀或分支中的条目。

当目录在多个服务器间划分时，每个服务器仅负责目录树的一个部分。分布式目录服务的工作方式与域名服务 (DNS) 类似，它将 DNS 命名空间的每个部分分配到特定的 DNS 服务器。同样，目录命名空间可以在服务器间分发，同时维护一个从客户端的角度来说的目录服务，似乎是单个目录树。

目录服务器还提供 知识参考，用于连接存储在不同数据库中的目录数据的机制。目录服务器包含两种知识参考：referrals 和 chaining。

本章的剩余部分描述了数据库和知识引用，介绍了两种知识参考类型之间的区别，并描述了如何设计索引以提高数据库的性能。

6.2. 分发目录数据

分发数据后，可以在多台服务器上扩展目录服务，无需实际地包含企业服务器中的每个服务器上的目录条目。因此，分布式目录可以保存更多的条目数量，而不是单个服务器可能。

另外，可将目录服务配置为隐藏用户的分发详情。在关注用户和应用程序的情况下，只有单个目录可以回答其目录查询。

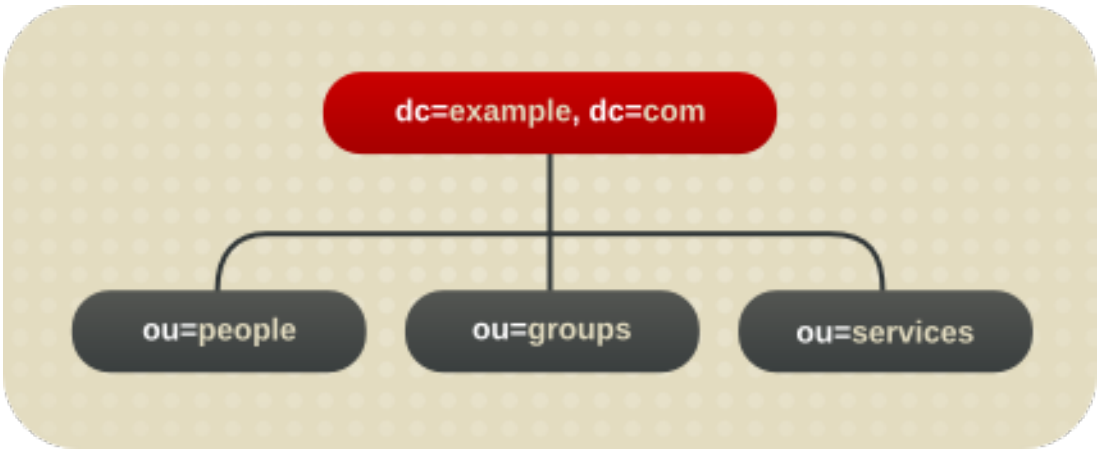
以下小节更详细地描述了数据分发的原理：

- 第 6.2.1 节 “关于使用多个数据库”
- 第 6.2.2 节 “关于 Suffixes”

6.2.1. 关于使用多个数据库

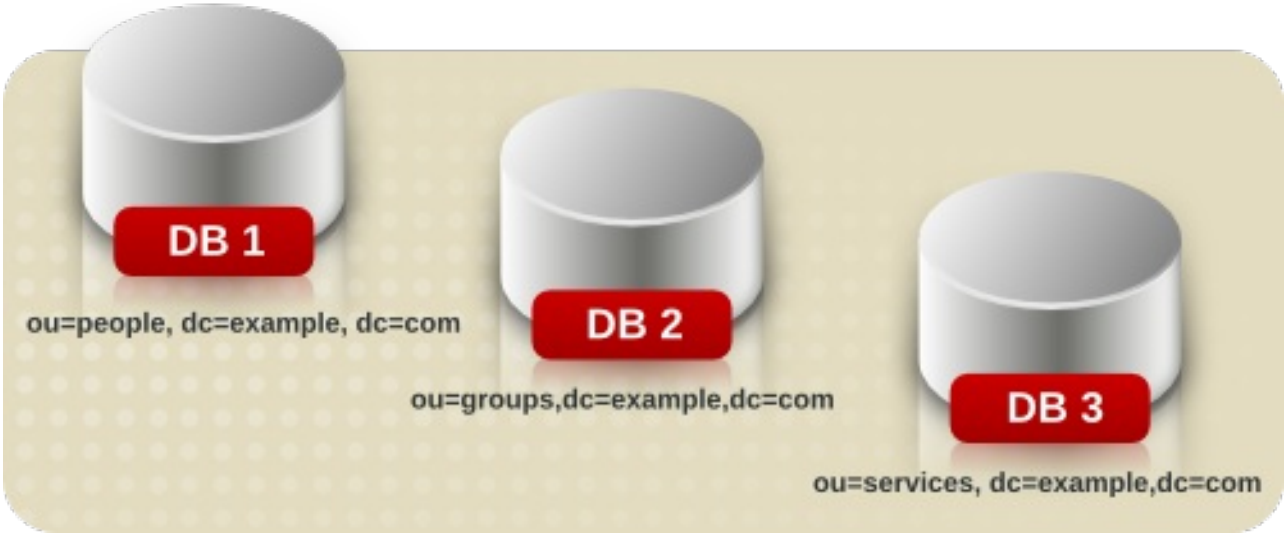
目录服务器将数据保存到 LDBM 数据库中。这个基于磁盘的高性能数据库。每个数据库由一组大型文件组成，其中包含分配给它的所有数据。

目录树的不同部分可以存储在不同的数据库中。



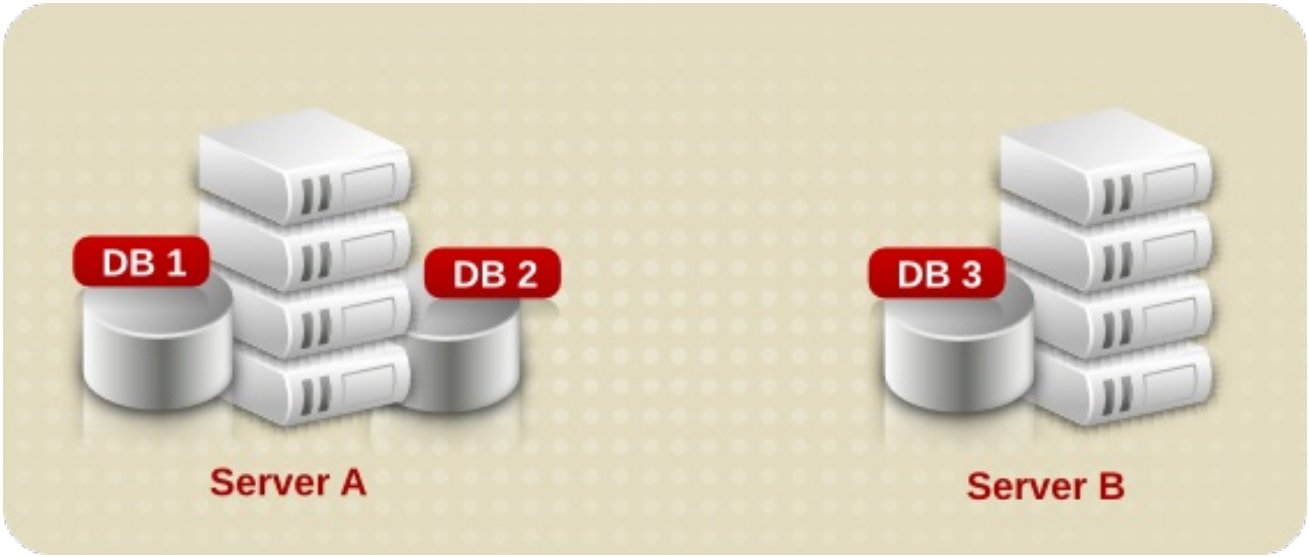
例如：图 6.1 “在独立的数据库中保存后缀数据” 显示存储在三个独立数据库中的三个后缀。

图 6.1. 在独立的数据库中保存后缀数据



当目录树划分到多个数据库时，可以在多个服务器上分发这些数据库。例如，如果存在三个数据库，DB1、DB2 和 DB3 含有目录树的三个后缀，则可以将其存储在两个服务器上，即 Server A 和 Server B。

图 6.2. 在独立的服务器间划分后缀数据库



服务器 A 包含 DB1 和 DB2，服务器 B 包含 DB3。

在多个服务器间分布数据库可减少每台服务器上的工作负载。因此，目录服务可以扩展到比单个服务器可能更多的条目。

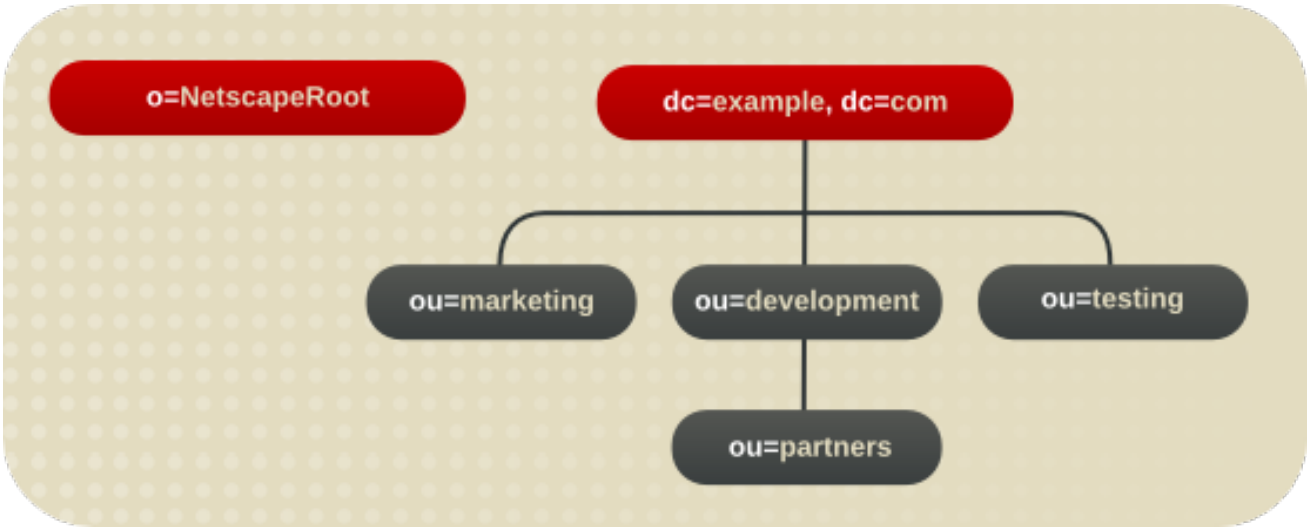
此外，Directory 服务器支持动态添加数据库，这意味着当目录服务在不需要整个目录服务的情况下，可以添加新的数据库。

6.2.2. 关于 Suffixes

每个数据库都包含 Directory 服务器的特定后缀中的数据。可以创建根和子修复来组织目录树的内容。root 后缀是树顶部的条目。它可以是目录树的根目录，也可以是为 Directory 服务器设计的更大树的一部分。subsuffix 是根后缀下的分支。root 和 subsuffixes 的数据包含在数据库中。

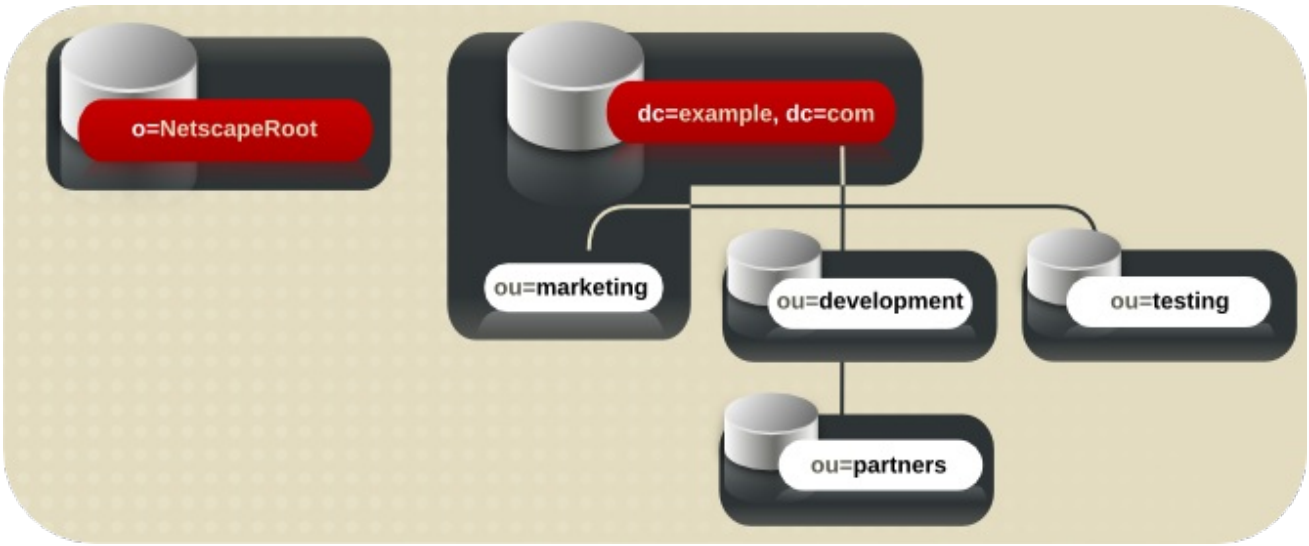
例如，example Corp. 创建后缀来代表目录数据的分布。

图 6.3. 示例公司的目录树.



示例 Corp. 可以将其目录树分散到五个不同的数据库中，如 图 6.4 "跨多个数据库进行目录树读取" 中。

图 6.4. 跨多个数据库进行目录树读取



生成的后缀会包含以下条目：

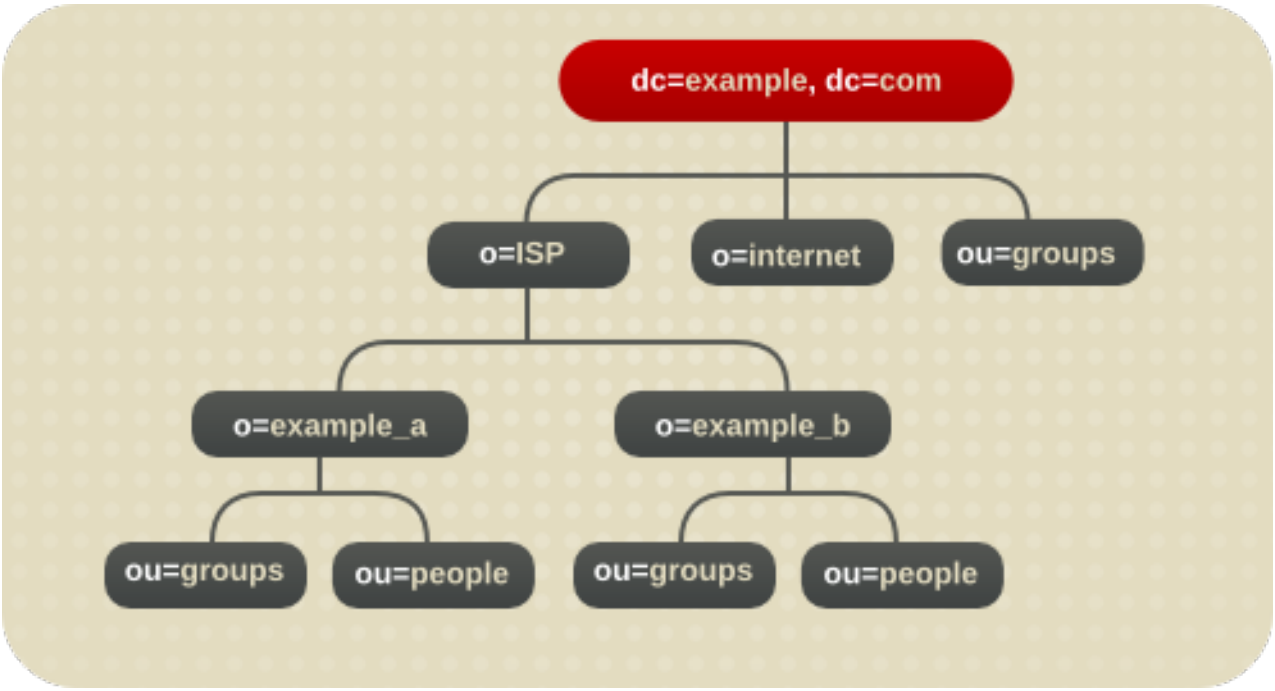
图 6.5. 分布式目录树的后缀

dc=example,dc=com 后缀是 root 后缀。ou=testing,dc=example,dc=com 后缀、ou=development,dc=example,dc=com 后缀和 ou=partners,ou=development,dc=example,dc=com 后缀是 dc=example,dc=com root 后缀的所有子后缀。root 后缀 dc=example,dc=com 包含原始目录树的 ou= marketing 分支中的数据。

使用多个 Root Suffixes

目录服务可以包含多个根后缀。例如，名为"Example"的 ISP 可以托管几个网站，一个用于 example_a.com，一个用于 example_b.com。ISP 将创建两个 root 后缀，一个用于 o=example_a.com 命名上下文，另一个对应于 o=example_b.com 命名上下文。

图 6.6. 带有多个 Root Suffixes 的目录树



dc=example,dc=com 条目代表 root 后缀。每个托管客户的条目也是一个 root 后缀(o=example_a 和 o=example_b)。ou=people 和 ou=groups 分支是每个 root 后缀下的子后缀。

6.3. 关于知识库参考

在将数据分发到多个数据库后，使用 知识引用 定义分布式数据之间的关系，指向不同数据库中保存的目录信息的指针。目录服务器提供以下知识引用，以帮助将分布式数据链接到单个目录树：

- *引用 - 服务器将一类信息返回到客户端应用，指示客户端应用程序需要联系另一个服务器来满足该请求。*

- 链 - 服务器代表客户端应用程序联系其他服务器，并在操作完成后将结果返回给客户端应用程序。

以下小节更详细地描述了并比较这两种类型的知识参考。

6.3.1. 使用引用

参照 (referral) 一个服务器返回的信息片段会告知客户端应用程序要联系以继续操作请求。当客户端应用程序请求本地服务器上不存在的目录条目时，会发生此重定向机制。

目录服务器支持两种类型的引用：

- 默认引用 - 当客户端应用程序显示服务器没有匹配后缀的 DN 时，目录会返回默认引用。默认引用存储在服务器的配置文件中。可以为 Directory Server 设置一个默认引用，每个数据库都有一个默认的引用。

  每个数据库的默认引用是通过后缀配置信息进行的。当禁用数据库的后缀时，将目录服务配置为将默认引用返回到向该后缀发出的客户端请求。

  有关后缀的详情，请参考 第 6.2.2 节 “关于 Suffixes”。有关配置后缀的详情，请参考 Red Hat Directory Server Administration Guide 。

- 智能引用 - 智能引用存储在目录服务本身内的条目上。智能引用指向目录服务器，其了解其 DN 与包含智能引用的条目的 DN 匹配。

所有引用均以 LDAP 统一资源 locator 或 LDAP URL 的格式返回。以下小节描述了 LDAP 引用的结构，然后描述目录服务器支持的两种引用类型。

6.3.1.1. LDAP 推荐结构

LDAP 引用包含 LDAP URL 格式的信息。LDAP URL 包含以下信息：

- 要联系的服务器的主机名。
- 配置为侦听 LDAP 请求的服务器上的端口号。
- 基本 DN（用于搜索操作）或目标 DN（用于添加、删除和修改操作）。

例如，客户端应用搜索 dc=example,dc=com 以查找具有 surname 值 Jensen 的条目。引用将以下 LDAP URL 返回给客户端应用程序：

> ldap://europe.example.com:389/ou=people, l=europe,dc=example,dc=com

此引用指示客户端应用程序联系端口 389 上的主机 europe.example.com，并使用根后缀 ou=people, l=europe,dc=example,dc=com 提交搜索。

LDAP 客户端应用程序决定如何处理引用。有些客户端应用程序会自动重试服务器中的操作。其他客户端应用程序会将参考信息返回给用户。Red Hat Directory Server （如命令行实用程序）提供的大多数 LDAP 客户端应用程序会自动遵循引用。初始目录请求中提供的相同绑定凭证用于访问服务器。

大多数客户端应用程序都遵循有限数量的引用或跃点。后面的引用数量的限制减少了客户端应用程序试图完成目录查找请求的时间，并有助于消除由环形引用模式导致的挂起进程。

6.3.1.2. 关于默认引用

当联系的服务器或数据库不包含请求的数据时，默认引用将返回到客户端。

目录服务器通过比较所请求目录对象的 DN 与本地服务器支持的目录后缀进行比较，以确定是否返回默认引用。如果 DN 与支持的后缀不匹配，目录服务器会返回一个默认的引用。

例如，目录客户端请求以下目录条目： uid=bjensen,ou=people,dc=example,dc=com

但是，服务器仅管理 dc=europe,dc=example,dc=com 后缀下存储的条目。目录会返回客户端，指示 dc=example,dc=com 后缀下存储的条目要联系哪个服务器。然后，客户端联系适当的服务器并重新提交原始请求。

将默认引用配置为指向包含目录服务分发的更多信息的目录服务器。服务器的默认引用由 nsslapd-referral 属性设置。目录中每个数据库的默认引用由配置中的数据库条目中的 nsslapd-referral 属性设置。这些属性值存储在 dse.ldif 文件中。

有关配置默认引用的详情，请参考 Red Hat Directory Server Administration Guide。

6.3.1.3. 智能引用

目录服务器也可以使用 智能引用。智能引用将目录条目或目录树关联到特定的 LDAP URL。这意味着请求可以转发到以下任意一种：

- 不同服务器上包含的同一命名空间。
- 本地服务器上的不同命名空间。
- 同一服务器上的不同命名空间。

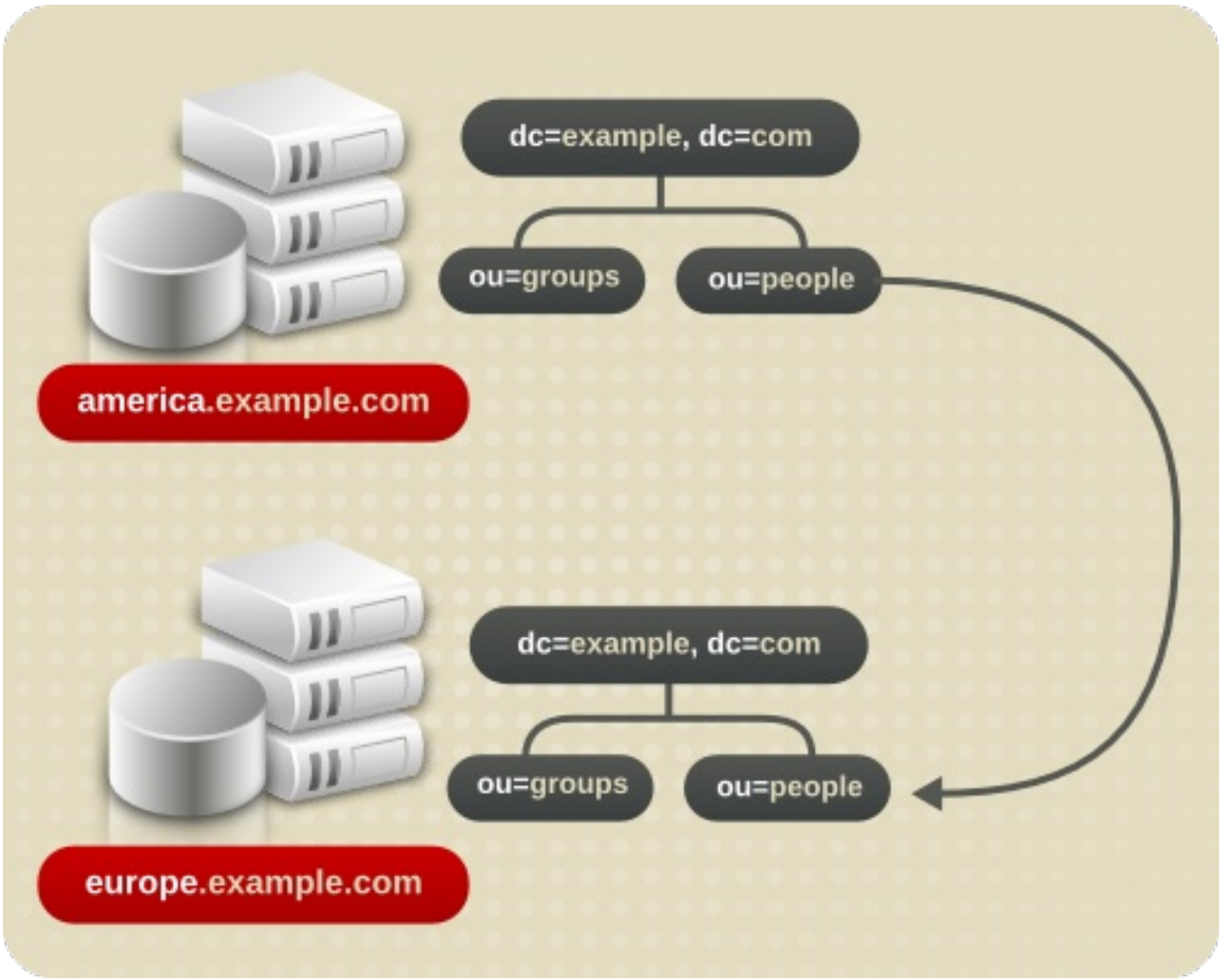
与默认引用不同，智能引用存储在目录服务本身中。有关配置和管理智能引用的详情，请参考 红帽目录服务器管理指南。

例如，Example Corp. 的美国办公室的目录服务包含 ou=people,dc=example,dc=com 目录分支点。

通过指定 ou=people 条目本身的智能引用，将此分支上的所有请求重定向到示例公司欧洲办事处的 ou=people 分支。智能引用是 ldap://europe.example.com:389/ou=people,dc=example,dc=com。

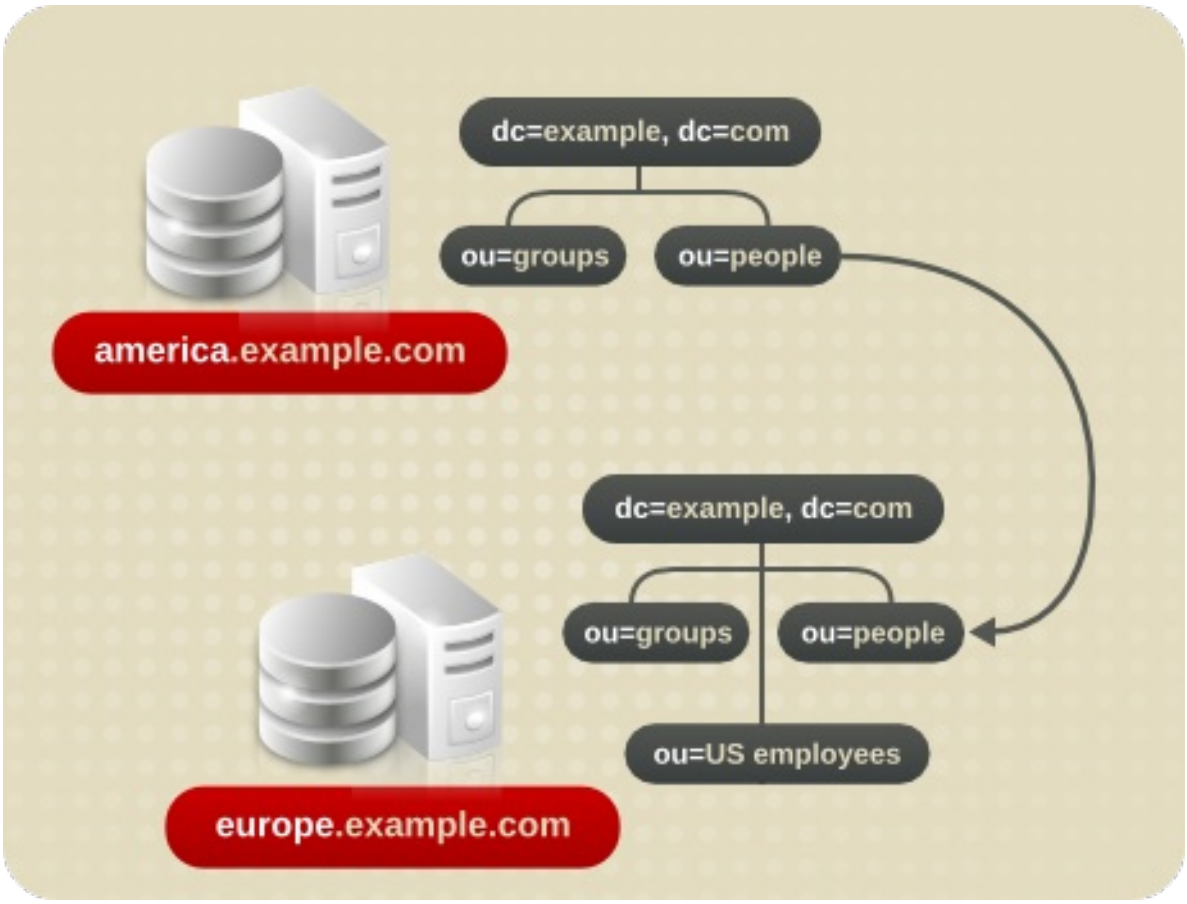
对 America 目录服务的人员分支提出的任何请求都将重定向到欧洲目录。下面是以下说明：

图 6.7. 使用智能引用重定向请求



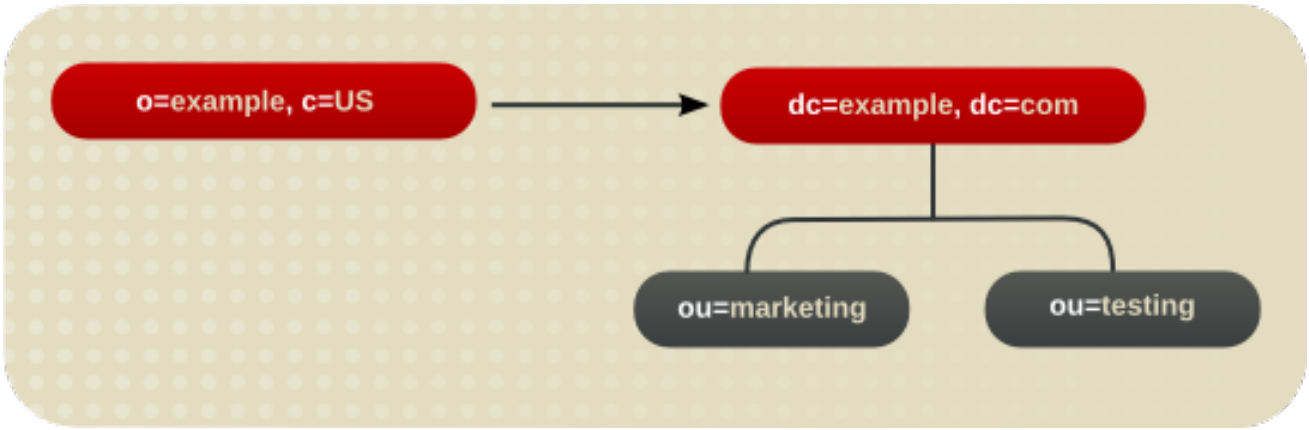
相同的机制可用于将查询重定向到使用不同的命名空间的不同服务器。例如，一个员工在意大利语办事处的 Example Corp. 工作。请针对美国示例 Corp. 员工的电话号码向欧洲目录服务提出请求。目录服务返回引用 ldap://europe.example.com:389/ou=US 员工，dc=example,dc=com。

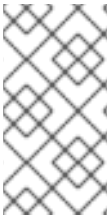
图 6.8. 将查询重定向到不同的服务器和客户端



最后，如果在同一服务器上提供了多个后缀，可以将查询重定向到在同一计算机上提供的另一个命名空间。例如，要将本地计算机上的所有查询重定向到 o=example,c=us 到 dc=example,dc=com，然后将 smart referral ldap:///dc=example,dc=com 放置到 o=example,c=us 条目。

图 6.9. 将一个命名空间中的 Query 重定向到 Same 服务器上的Another Namespace





注意

此 LDAP URL 中的第三斜杠表示 URL 指向同一目录服务器。

从一个命名空间创建引用仅针对其搜索以区分名称的客户端使用。其他类型的操作（如 ou=people,o=example,c=US）没有被正确执行。

有关 LDAP URLS 以及如何在目录服务器条目中包含智能 URL 的更多信息，请参阅 红帽目录服务器管理指南。

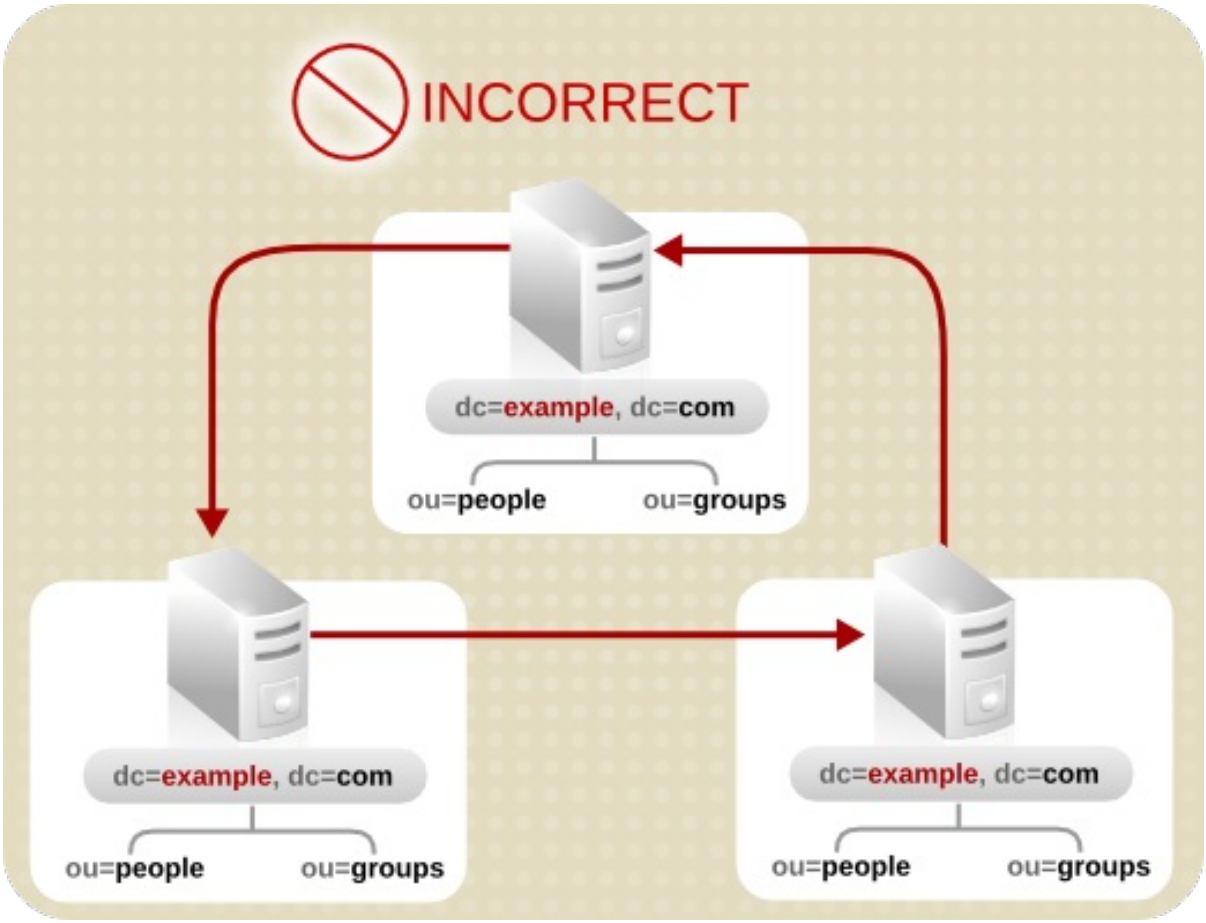
6.3.1.4. 设计智能清单的提示

尽管智能引用易于实现，但在使用前请考虑以下点：

- 简化设计。

  使用复杂的 Web 部署目录服务会导致管理困难。过度使用智能引用也可以导致循环引用模式。例如，一个引用指向 LDAP URL，该 URL 又指向另一个 LDAP URL，以此类推，直到链中的参考位置返回原始服务器。下面是以下说明：

  图 6.10. Circular 推荐模式



- 重定向位于主要分支点.

  限制引用使用，以便在目录树的后缀级别处理重定向。智能引用将 leaf(non-branch)条目的查找请求重定向到不同的服务器和 DN。因此，它会临时使用 smart 引用作为别名机制，从而导致出现一种复杂而困难的方法来保护目录结构。将引用限制为目录树的后缀或主要分支点，限制了必须管理的引用数量，从而减少了目录的管理开销。

- 考虑安全性影响。

  访问控制不可跨越引用。即使源自该请求的服务器允许访问条目，智能引用时也向其他服务器发送客户端请求，但客户端应用可能无法被允许访问。

  此外，客户端的凭据需要在服务器上可用，上面提到了客户端进行客户端身份验证。

6.3.2. 使用链

链(Chaining)是一种将请求中继到其他服务器的方法。此方法通过数据库链接实现。数据库链接，如第 6.2 节 "分发目录数据" 所述，不包含任何数据。相反，它会将客户端应用程序请求重定向到包含数据的远程服务器。

在串联过程中，服务器从客户端应用程序收到服务器不包含的数据的请求。使用数据库链接时，服务器会代表客户端应用程序联系其他服务器，并将结果返回到客户端应用。

每个数据库链接都与保留数据的远程服务器关联。配置包含数据库链接复制的备用远程服务器，以便在出现故障时使用。有关配置数据库链接的更多信息，请参阅 红帽目录服务器管理指南。

数据库链接提供以下功能：

- 不允许访问远程数据。

  由于数据库链接可以解决客户端请求，因此数据分布在客户端中完全隐藏。

- 动态管理。

  可在整个系统一直供客户端应用程序使用时，在系统中添加或删除目录服务的一部分。数据库链接可以临时返回引用到应用程序，直到在目录服务中重新分发条目。

  这也可以通过后缀本身实现，它可以返回引用而不是将客户端应用程序转发到数据库。

- 访问控制.

  数据库链接模拟客户端应用，为远程服务器提供适当的授权身份。当不需要访问控制评估时，可以在远程服务器上禁用用户模仿。有关配置数据库链接的更多信息，请参阅 红帽目录服务器管理指南。

6.3.3. 在引用和链之间决定

连接目录分区的方法都各有优缺点。要使用的方法或方法组合取决于目录服务的特定需求。

两个知识引用之间的主要区别在于了解如何找到分布式信息的情报位置。在连锁系统中，在服务器中实施智能功能。在使用引用的系统中，智能在客户端应用程序中实施。

串联会降低客户端的复杂性，但会降低服务器复杂性。链的服务器必须与远程服务器一起使用，并将结果发送到目录客户端。

使用引用时，客户端必须处理查找引用和整合搜索结果。但是，参考者为客户端应用程序提供了更大的灵活性，并允许开发人员在分布式目录操作中为用户提供更好的反馈。

以下小节描述了在更详细引用和链接之间的一些更具体的不同之处。

6.3.3.1. 使用差异

有些客户端应用程序不支持引用。链允许客户端应用程序与单一服务器通信，并仍然访问存储在多个服务器上的数据。有时，当公司的网络使用代理时，引用无法正常工作。例如，客户端应用可能具有仅与防火墙中的一个服务器通信的权限。如果该应用程序被称为其他服务器，则无法成功联系它。

在使用引用时，客户端还必须能够正确进行身份验证，这意味着要引用客户端的服务器需要包含客户端的凭据。使用链时，客户端身份验证仅会出现一次。客户端不需要在将请求链的服务器上再次进行身份验证。

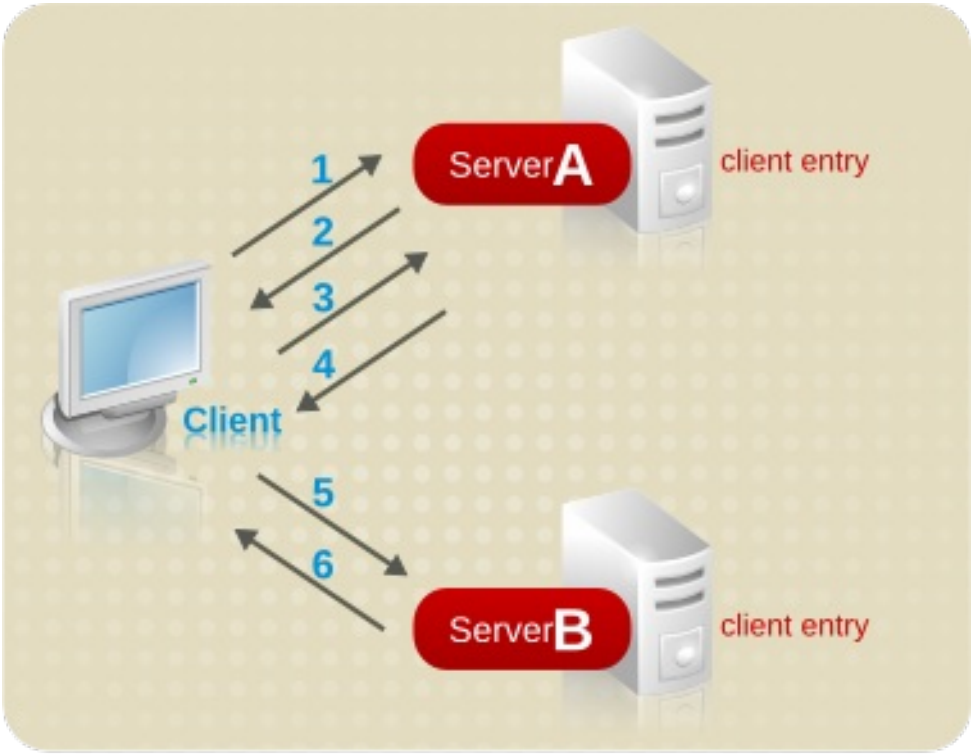
6.3.3.2. 评估访问控制

链评估访问控制与引用不同。使用引用时，客户端的条目必须在所有目标服务器上存在。对于链，客户端条目不需要在所有目标服务器上。

使用引用执行搜索请求

下图显示了使用参考到服务器的客户端请求：

图 6.11. 使用引用将客户端请求发送到服务器



在以上说明中，客户端应用程序执行以下步骤：

1. 客户端应用首先与服务器 A 绑定。
2. server A 包含一个用于提供用户名和密码的客户端的条目，因此它会返回一个 bind 接受消息。为使推荐工作，客户端条目必须在 server A 上显示。
3. 客户端应用将操作请求发送到服务器 A。
4. 但是，Server A 不包含请求的信息。相反，服务器 A 会向客户端应用程序返回一个引用，指示它联系服务器 B。

5. 然后，客户端应用程序会向 Server B 发送绑定请求。要成功绑定，服务器 B 还必须包含客户端应用程序的条目。

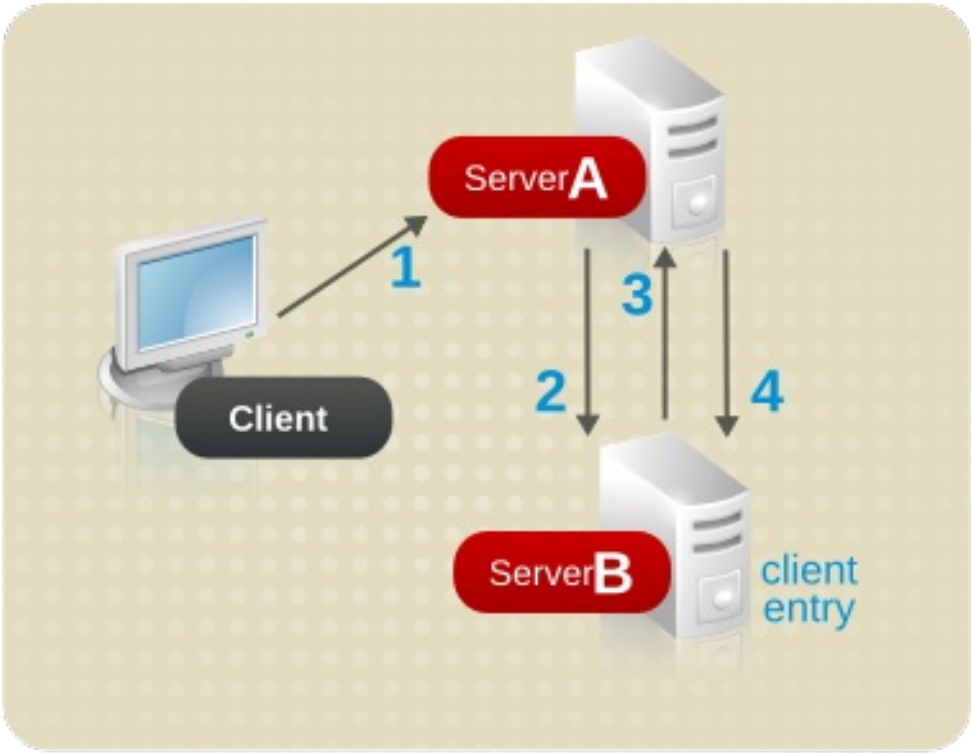
6. 绑定成功，客户端应用程序现在可以将其搜索操作重新提交到 Server B。

这个方法要求 Server B 从 Server A 中包含客户端条目的复制副本。

使用链执行搜索请求

使用链解决跨服务器复制客户端条目的问题。在链的系统上，搜索请求会多次转发，直到有响应为止。

图 6.12. 使用链将客户端请求发送到服务器



在以上说明中，执行以下步骤：

1. 客户端应用与服务器 A 绑定，服务器 A 会尝试确认用户名和密码是否正确。

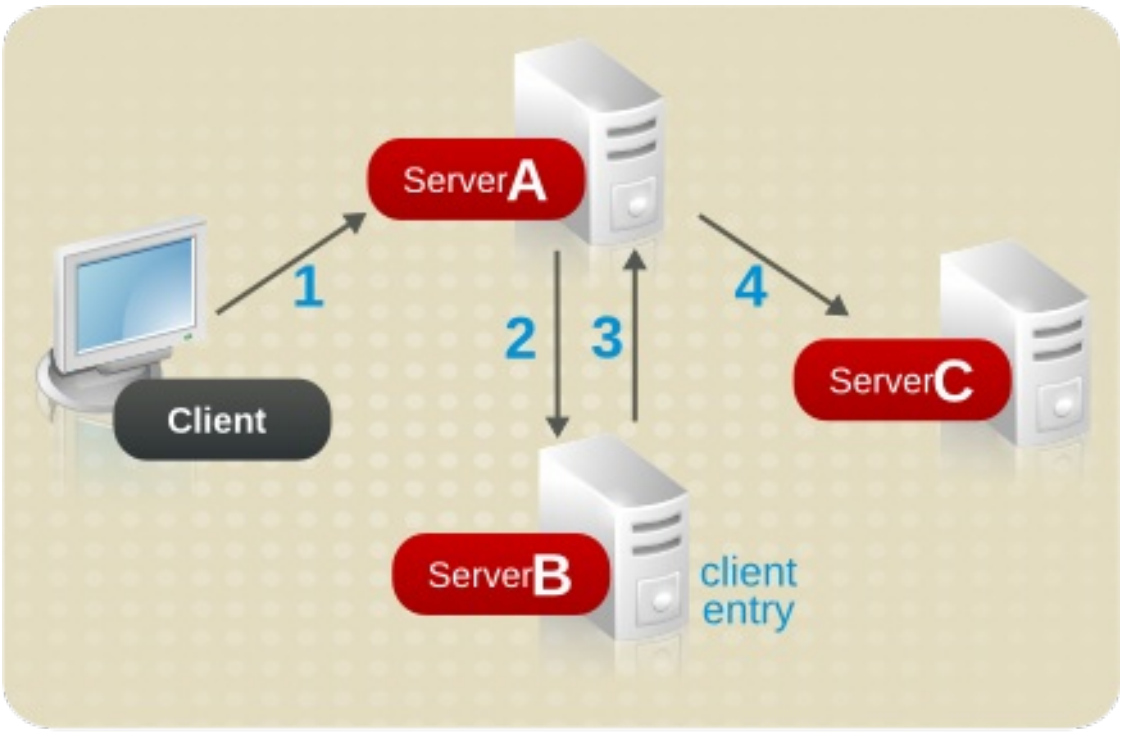
2. 服务器 A 不包含与客户端应用程序对应的条目。相反，它包含一个到 Server B 的数据库链接，其中包含客户端的实际条目。服务器 A 将绑定请求发送到服务器 B。

3. 服务器 B 将接受响应发送到服务器 A。

4. 然后，服务器 A 使用数据库链接处理客户端应用的请求。数据库链接联系位于 Server B 上的远程数据存储，以处理搜索操作。

在链系统中，与客户端应用程序对应的条目不需要位于与客户端请求的数据相同的服务器上。

图 6.13. 使用不同服务器验证客户端和检索数据



在这个图示中会执行以下步骤：

1. 客户端应用与服务器 A 绑定，服务器 A 会尝试确认用户名和密码是否正确。
2. 服务器 A 不包含与客户端应用程序对应的条目。相反，它包含一个到 Server B 的数据库链接，其中包含客户端的实际条目。服务器 A 将绑定请求发送到服务器 B。
3. 服务器 B 将接受响应发送到服务器 A。
4. 然后，服务器 A 使用另一个数据库链接处理客户端应用的请求。数据库链接联系位于 Server C 上的远程数据存储，以处理搜索操作。

不支持的访问控制

数据库链接不支持以下访问控制：

- 当用户条目位于其他服务器上时，不支持访问用户条目的内容。这包括基于组、过滤器和角色的访问控制。
- 可能会拒绝基于客户端 IP 地址或 DNS 域的控制。这是因为数据库链接在联系远程服务器时模拟客户端。如果远程数据库包含基于 IP 的访问控制，它会使用数据库链接的域而不是原始客户端域来评估它们。

6.4. 使用索引来提升数据库性能

根据数据库的大小，客户端应用程序执行的搜索可以是时间和资源密集型。为了帮助缓解这个问题，请使用索引来提高搜索性能。

索引是存储在目录数据库中的文件。为目录服务中的每个数据库维护单独的索引文件。每个文件根据其索引的属性命名。特定属性的索引文件可以包含多个类型的索引，因此可以为每个属性维护多种类型的索引。例如，名为 **cn.db** 的文件包含通用 name 属性的所有索引。

根据使用目录服务的应用程序类型，使用不同类型的索引。不同的应用可能会经常搜索特定属性，或者可能会以不同语言搜索目录，或者可能需要特定格式的数据。

6.4.1. 目录索引类型概述

目录服务器支持以下索引类型：

- presence index - 列出具有特定属性的条目，如 uid。
- equality index - 列出包含特定属性值的条目，如 cn=Babs Jensen。
- 大约索引 - 允许近似（或"类似于"的"）搜索。例如，条目可能包含 cn=Babs L. Jensen 的属性值。大约搜索将返回针对 cn~=Babs Jensen、cn~=Babs 和 cn~=Jensen 进行搜索的值。

注意

大约索引要求使用 ASCII 字符以英文写入名称。

- substring index - 允许搜索条目内的子字符串。例如，搜索 cncategoriesderson 将匹配包含此字符串的通用名称（如 Bill Anderson、Norma Henderson 和 netobserv Sanderson）。
- 国际索引 - 提高了搜索在国际目录中信息的性能。通过将区域设置(internationalization OID)与要索引的属性关联，将索引配置为应用匹配的规则。
- 浏览索引或虚拟列表视图(VLV)索引 - 提高了 web 控制台中条目的显示性能。可以在目录树中的任何分支中创建浏览索引，以提高显示性能。

6.4.2. 评估索引的成本

索引提高了目录数据库中的搜索性能，但成本涉及到：

- 索引会增加修改条目所需的时间。

  维护的索引越长，目录服务需要更新数据库所需的时间。

- 索引文件使用磁盘空间。

  使用更多属性进行索引，会创建更多文件。如果包含长字符串的属性有大约和子字符串的索引，则这些文件可能会快速增长。

- 索引文件使用内存。

  要更有效地运行，目录服务会将尽可能多的索引文件放在内存中。根据数据库缓存大小，索引文件使用池中可用的内存。大量索引文件需要更大的数据库缓存。

- 创建索引文件需要一些时间。

  虽然索引文件在搜索过程中节省时间，但维护不必要的索引会浪费时间。请确保仅使用目录服务维护客户端应用程序所需的文件。

第 7 章 设计复制过程

复制目录的内容会增加目录服务的可用性和性能。第 4 章 设计目录树 第 6 章 设计目录拓扑 涵盖目录树和目录拓扑的设计。本章解决了数据的物理和地理位置，特别是如何使用复制来确保数据在和需要时可用。

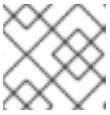
本章讨论使用复制，并提供有关为目录环境设计复制策略的建议。

7.1. 复制简介

复制是自动将目录数据从一个 Red Hat Directory Server 复制到另一个的机制。使用复制时，任何目录树或子树（存储在自己的数据库中）都可在服务器间复制。保存信息的主副本的目录服务器会自动将任何更新复制到所有副本。

复制提供高可用性目录服务，并可分布地理的数据。在实际术语中，复制具有以下优点：

- 容错和故障转移 - 通过将目录树复制到多个服务器，即使硬件、软件或网络问题也提供了目录服务，即使是硬件、软件或网络问题也会阻止目录客户端应用程序访问特定的目录服务器。客户端被称为另一个目录服务器进行读写操作。

  注意

  写入故障切换只能在多层次复制的情况下实现。

- 负载平衡 - 在服务器间复制目录树可减少对任何给定计算机的访问负载，从而改进了服务器响应时间。
- 更高的性能和减少响应时间 - 在用户接近的位置复制目录条目可显著改进了目录响应时间。
- 本地数据管理 - 复制允许本地拥有和管理信息，同时与整个企业的其他目录服务器共享。

7.1.1. 复制概念

根据以下基本决策，开始规划复制：

- 要复制哪些信息。
- 哪些服务器拥有该信息的主副本，或 读写副本。
- 哪些服务器拥有该信息的只读副本，或 只读副本。
- 当只读副本收到更新请求时，应该会出现什么情况，即它应该引用该请求的服务器。

在了解目录服务器如何处理这些概念的情况下，无法有效地进行这些决策。例如，决定要复制哪些信息，请注意目录服务器可以处理的最小复制单元。Directory 服务器使用的复制概念提供了一个框架，用于考虑需要做出的全局决策。

7.1.1.1. 复制单元

复制的最小单元是数据库。整个数据库可以复制，但不能在数据库中复制子树。因此，在定义目录树时，总是考虑复制。有关如何设置目录树的详情请参考 第 4 章 设计目录树。

复制机制还需要一个数据库与一个后缀对应。超过两个或多个数据库的后缀（或命名空间）无法复制。

7.1.1.2. 读写和只读副本

参与复制的数据库被定义为副本。目录服务器支持两种类型的副本：读写和只读。读写副本包含目录信息的主副本，可以更新。只读副本引用对读写副本的所有更新操作。

7.1.1.3. 供应商和消费者

存储复制到不同服务器的副本的服务器称为 供应商。存储从不同服务器复制的副本的服务器称为 消费者。通常说，供应商服务器上的副本是一个读写副本，使用者服务器上的副本是一个只读副本。但是，会有以下例外：

- 对于 级联复制，hub 供应商 会包含它提供给消费者的只读副本。更多信息请参阅 第 7.2.3 节 “cascading Replication”。
- 对于multi-supplier replication，供应商可以将供应商和消费者用作同一读写副本的用户。更多信息请参阅 第 7.2.2 节 “Multi-Supplier Replication”。

注意

在当前版本的 Red Hat Directory Server 中，复制始终由供应商服务器启动，而不是由消费者启动。这与 Directory 服务器的早期版本不同，允许消费者发起复制（其中使用者服务器可以从供应商服务器检索数据）。

供应商

对于任何特定副本，供应商服务器必须：

- 响应从目录客户端读取请求和更新请求。
- 维护副本的状态信息和 changelog。
- 启动到消费者服务器的复制。

供应商服务器始终负责记录其管理的读写副本的更改，因此供应商服务器会确保将任何更改复制到消费者服务器。

消费者

消费者服务器必须：

- 响应读取请求。
- 请参阅对副本的供应商服务器更新请求。

每当消费者服务器收到添加、删除或更改条目的请求时，该请求都会被称为副本的供应商。供应商服务器执行请求，然后复制更改。

Hub 厂商

在级联复制的特殊情况下，hub 供应商必须：

- 响应读取请求。
- 请参阅对副本的供应商服务器更新请求。
- 启动到消费者服务器的复制。

有关级联复制的详情请参考 第 7.2.3 节 “cascading Replication”。

7.1.1.4. 复制和更改日志

每个供应商服务器均有 changelog。更改日志是副本中发生的修改记录。然后，供应商服务器会在消费者服务器上回放这些修改，或当多层次复制时在其他供应商上回放这些修改。

修改条目时，会在 changelog 中记录描述执行的 LDAP 操作的更改记录。

changelog 的大小使用两个属性维护，即 **nsslapd-changelogmaxage** 或 **nsslapd-changelogmaxentries**。这些属性会修剪旧的 changelogs，以合理的更改日志大小。

7.1.1.5. 复制协议

目录服务器使用复制协议来定义复制。复制协议描述了单个供应商和单一消费者之间的复制。该协议是在供应商服务器上配置的。它标识：

- 要复制的数据库。
- 将数据推送到的使用者服务器。
- 复制发生的时间。
- 供应商服务器必须使用的 DN 来绑定（称为 供应商绑定 DN）。
- 连接的保护方式（TLS、启动 TLS、客户端验证、SASL 或简单身份验证）。
- 任何不会被复制的属性（请参阅 第 7.3.2 节 “使用 Fractional Replication 复制复制所选属性”）。

7.1.2. 数据一致性

一致性指的是复制数据库的内容在给定时间点上如何相互匹配。在服务器间复制的部分是调度更新。供应商服务器始终决定消费者服务器何时需要更新并启动复制。

目录服务器提供在一周中的特定时间或一天保留副本始终同步或调度更新的选项。

保持副本持续同步的优点是它提供更好的数据一致性。成本是来自频繁更新操作的网络流量。这个解决方案是在以下情况下的最佳选择：

- 服务器之间有可靠、高速连接。
- 目录服务服务的客户端请求主要是搜索、读取和写入和比较操作，以及相对较少的更新操作。

如果数据一致性程度较低，请选择最适合网络模式的更新频率，或降低对网络流量的影响。有些情况下，有计划更新而不是持续更新是最佳解决方案：

- 可用网络连接不可靠或间歇性。
- 目录服务服务的客户端请求主要更新操作。
- 需要降低通信成本。

在多层次复制的情况下，每个供应商的副本都表示 松散一致，因为在任何给定时间，存储在每个供应商中的数据中可能存在差异。这的确如此，即使副本持续同步，原因如下：

- 在供应商间传播更新操作时会有一个延迟。
- 提供服务更新操作的供应商不会等待第二个供应商在向客户端返回"成功"消息前进行验证。

7.2. 常见复制场景

决定从服务器到服务器的更新流以及服务器在传播更新时如何进行交互。有四个基本场景和几个策略来决定适合环境的方法。这些基本场景可以合并，以构建最适合网络环境的复制拓扑。

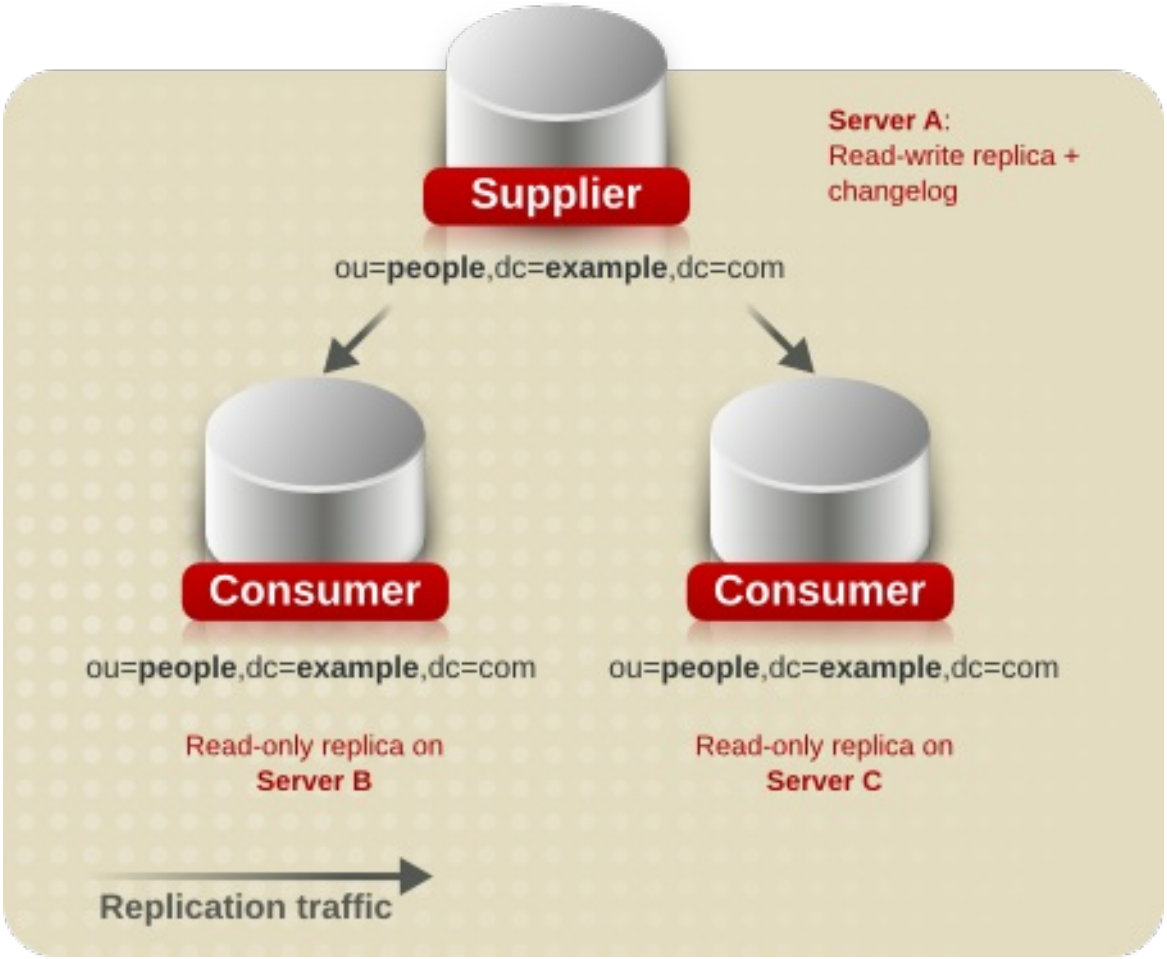
- *第 7.2.1 节 "单层复制"*
- *第 7.2.2 节 "Multi-Supplier Replication"*
- *第 7.2.3 节 "cascading Replication"*
- *第 7.2.4 节 "混合环境"*

7.2.1. 单层复制

在最基本的复制配置中，供应商服务器会将副本直接复制到一个或多个消费者服务器。在这个配置中，所有目录修改都会在供应商服务器上的读写副本中进行，消费者服务器会包含数据的只读副本。

供应商服务器必须对存储在消费者服务器上的读写副本执行所有修改。下面展示了这一点。

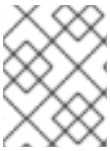
图 7.1. 单层复制



供应商服务器可以将读写副本复制到多个消费者服务器。单一供应商服务器可以管理的使用者服务器总数取决于网络的速度以及每天修改的条目总数。然而，供应商服务器能够维护多个消费者服务器。

7.2.2. Multi-Supplier Replication

在多层次复制环境中，同一信息的主副本可以存在于多个服务器上。这意味着可在不同的位置同时更新数据。每台服务器上发生的更改将复制到其他服务器上。这意味着每个服务器作为供应商和消费者的功能。



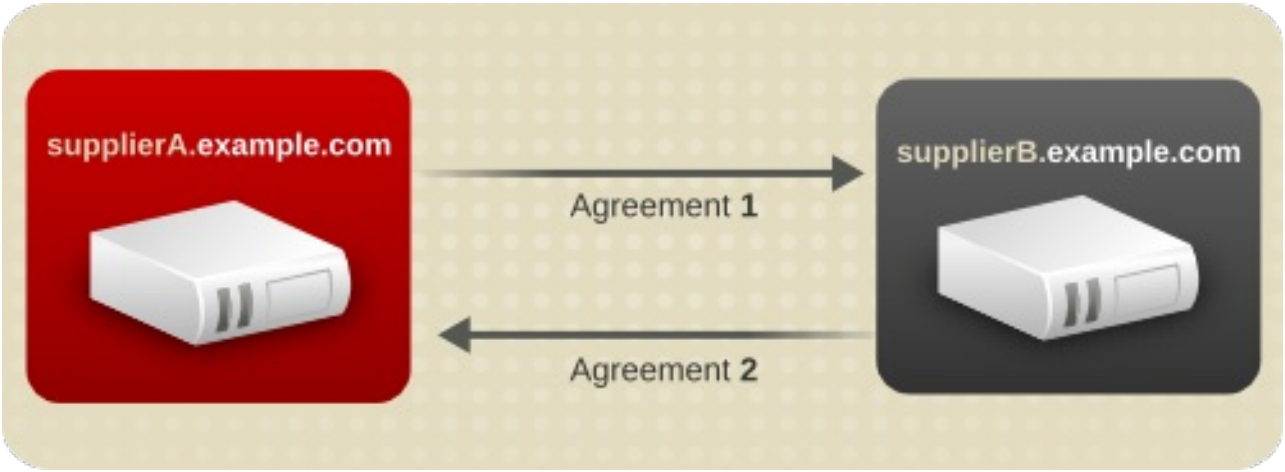
注意

Red Hat Directory Server 支持任何复制环境中最多 20 个供应商服务器，以及无限数量的 hub 供应商。拥有只读副本的使用者服务器数量是无限的。

当在多个服务器上修改相同数据时，会有一个冲突解析程序来确定要保留哪些更改。目录服务器将有效的更改视为最新的更改。

多个服务器可以有相同数据的主副本，但在单个复制协议范围内，只有一个供应商服务器和一个消费者。因此，要在两个供应商服务器间创建一个用于同一数据负责的多层环境，请创建多个复制协议。

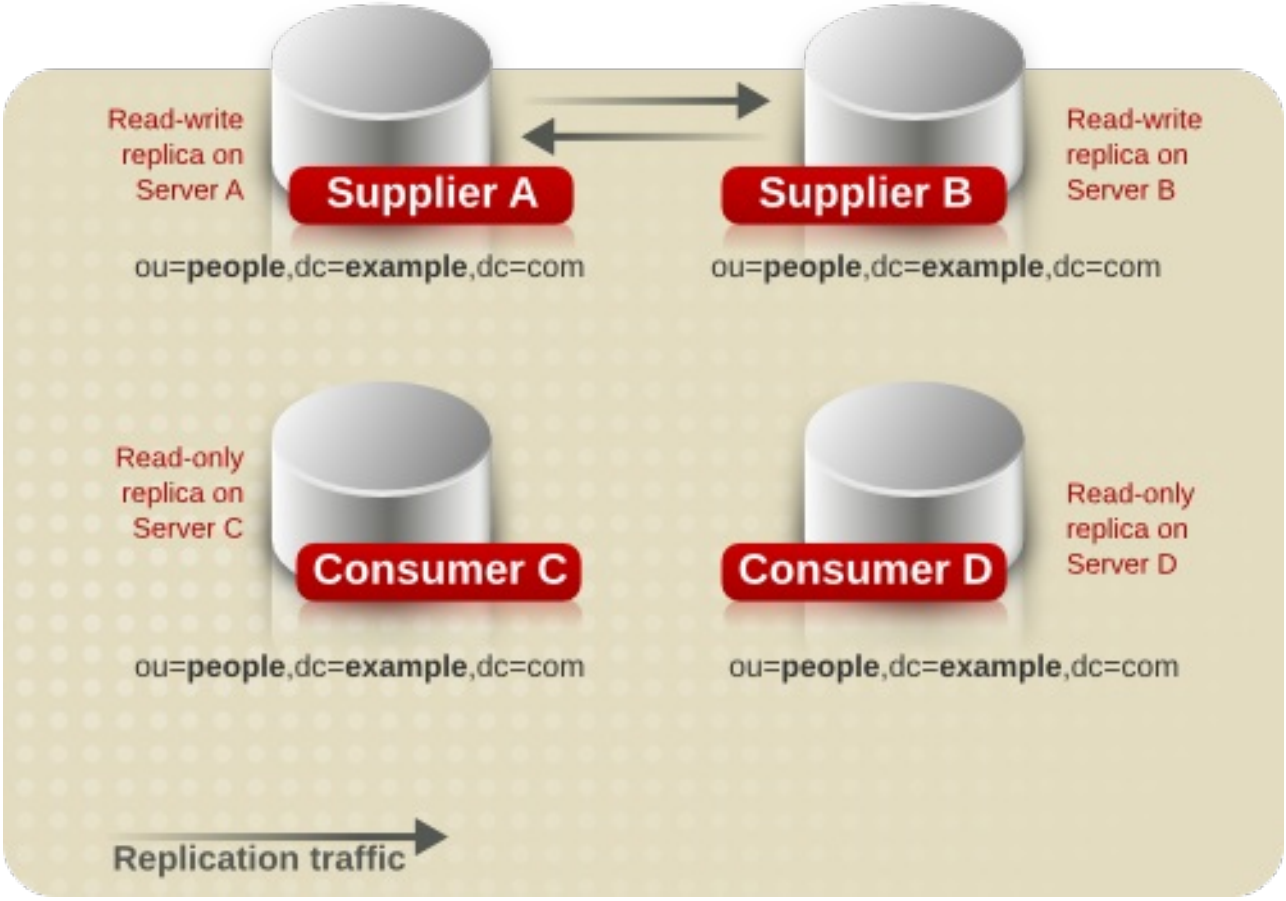
图 7.2. 简化多层次复制配置



在 图 7.2 “简化多层次复制配置” 中，供应商 A 和 provider B 各自包含同一数据的读写副本。

图 7.3 “在简单多Supplier 环境中复制流量” 演示了复制流量，其中包含两个供应商（图示中的读写副本）和两个消费者（图示中只读副本）。消费者可由两个供应商更新。供应商服务器可确保更改不会冲突。

图 7.3. 在简单多Supplier 环境中复制流量



目录服务器中的复制可以支持 20 个供应商，它们共享对相同数据的责任。使用许多供应商需要创建一系列复制协议。（也请记住，在多层次复制中，每个供应商都可以在不同拓扑中进行配置 - 即有 20 个不同的目录树甚至模式差异。有很多变量对拓扑选择直接影响。）

在多层次复制中，供应商可以将更新发送到所有其他供应商，或向其他供应商的一些子集发送更新。向所有其他供应商发送更新意味着更快地对更改进行请求，整个方案的整体方案具有更好的故障容错性。然而，它还增加了配置供应商的复杂性，并带来高网络需求和高服务器需求。向供应商子集发送更新要更简单地配置和减少网络和服务器负载，但数据在存在多个服务器故障时可能会丢失。

图 7.4 "多层复制配置 A" 演示了一个完全连接的网格拓扑，其中四个供应商服务器向其他三个供应商服务器提供数据（也作为消费者运作）。四家供应商服务器之间都存在一系列调整复制协议。

图 7.4. 多层复制配置 A

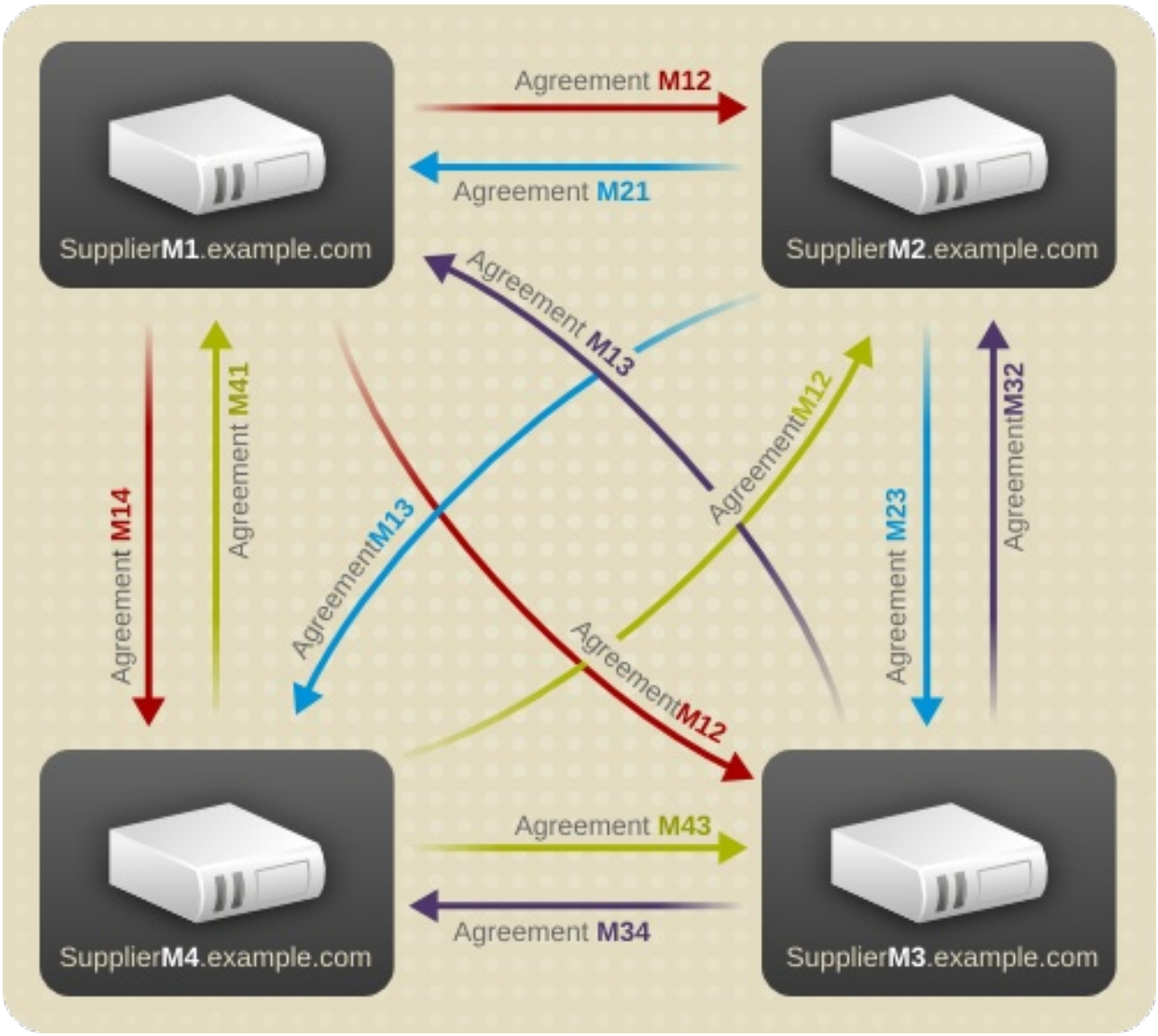
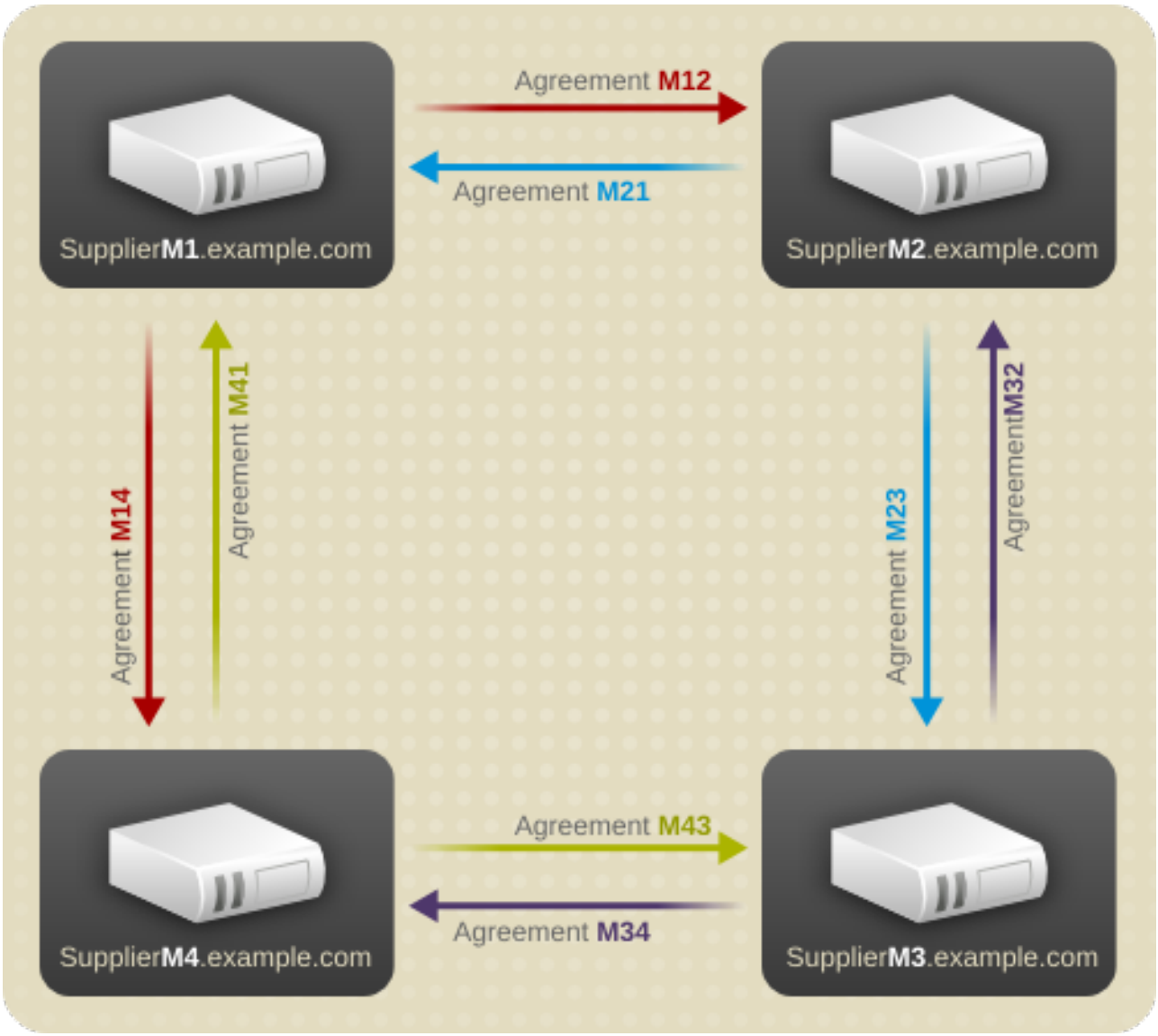


图 7.5 “配置 B” 演示了一个拓扑，即每个供应商服务器将数据发送到另外两个供应商服务器（也可以作为消费者使用）。与 图 7.4 “多层复制配置 A” 中的拓扑所示，只有 8 个复制协议存在于四个供应商服务器间，与 中拓扑显示的有十二协议。此拓扑很有用，有可能同时出现两个或更多个服务器失败。

图 7.5. 配置 B



这两示例就是简化的多层次方案。由于红帽目录服务器可以在单个多层次环境中拥有 20 多个供应商和数量无限数量的 hub 供应商，因此复制拓扑可能会变得更为复杂。例如：图 7.4 "多层复制配置 A" 有 12 个复制协议（我们的供应商各自有三个协议）。如果有 20 个供应商，则有 380 个复制协议（每个有 19 个协议的 20 个服务器）。

在规划多supplier 复制时，请考虑：

- *存在多少个供应商*
- *其地理位置*
- *供应商将用于更新其他位置的服务器的路径*
- *不同供应商的拓扑、目录树和模式*
- *网络质量*
- *服务器负载和性能*
- *目录数据所需的更新间隔*

7.2.3. cascading Replication

在级联复制方案中，hub 供应商从供应商服务器接收更新并在消费者服务器上回放这些更新。hub 供应商是一个混合的；它包含了只读副本，如典型的消费者服务器，它也像典型的供应商服务器一样维护一个更改日志。

当供应商从原始供应商接收供应商数据时，中心供应商会转发供应商数据。同样，当 hub 供应商从目录客户端收到更新请求时，它会引用供应商服务器的客户端。

如果组织内不同位置之间的某些网络连接比其他位置相比，级联复制非常有用。例如，Example Corp. 将其目录数据的主副本保存在 Minneapolis 中，以及 New York 和 Chicago 中的消费者服务器。Minneapolis 和 New York 之间的网络连接非常好，但 Minneapolis 和 Chicago 之间的连接不佳。由于 New York 和 Chicago 之间的网络是公平的，示例管理员使用 cascading replication 将目录数据从 Minneapolis 移到 New York to Chicago。

图 7.6. cascading Replication Scenario

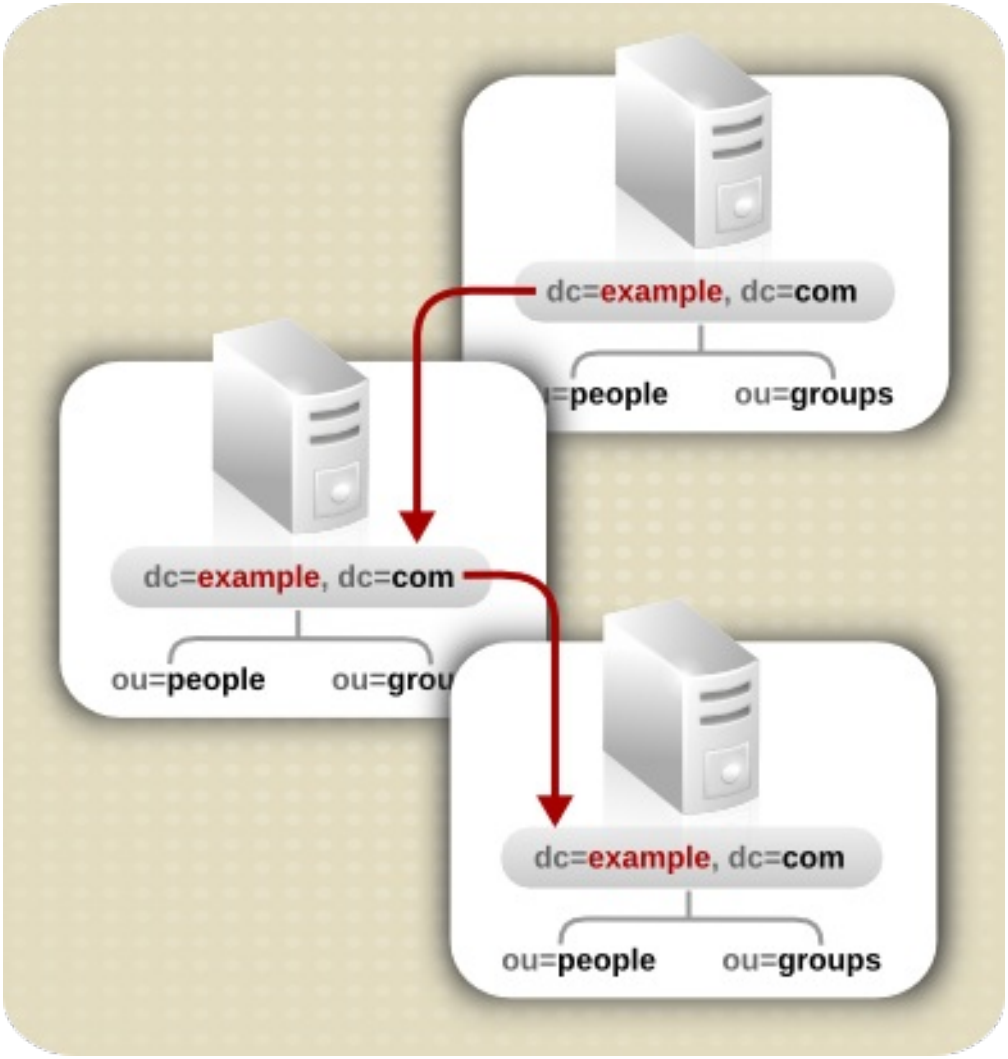
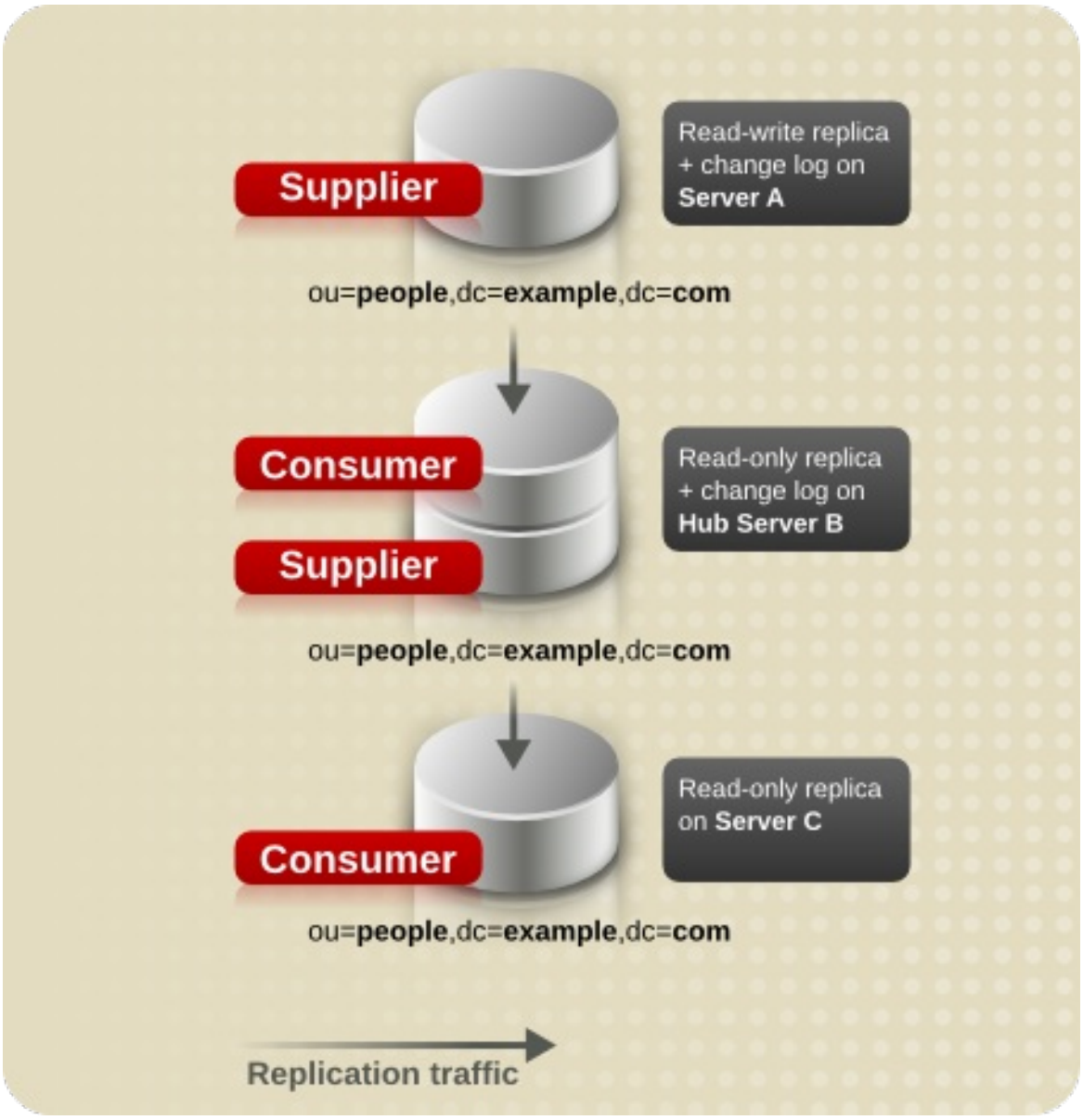


图 7.7 "复制流量和更改日志" 从不同的视角演示了相同的场景，它显示了如何在每台服务器上配置副本（读写或只读），以及哪些服务器维护更改日志。

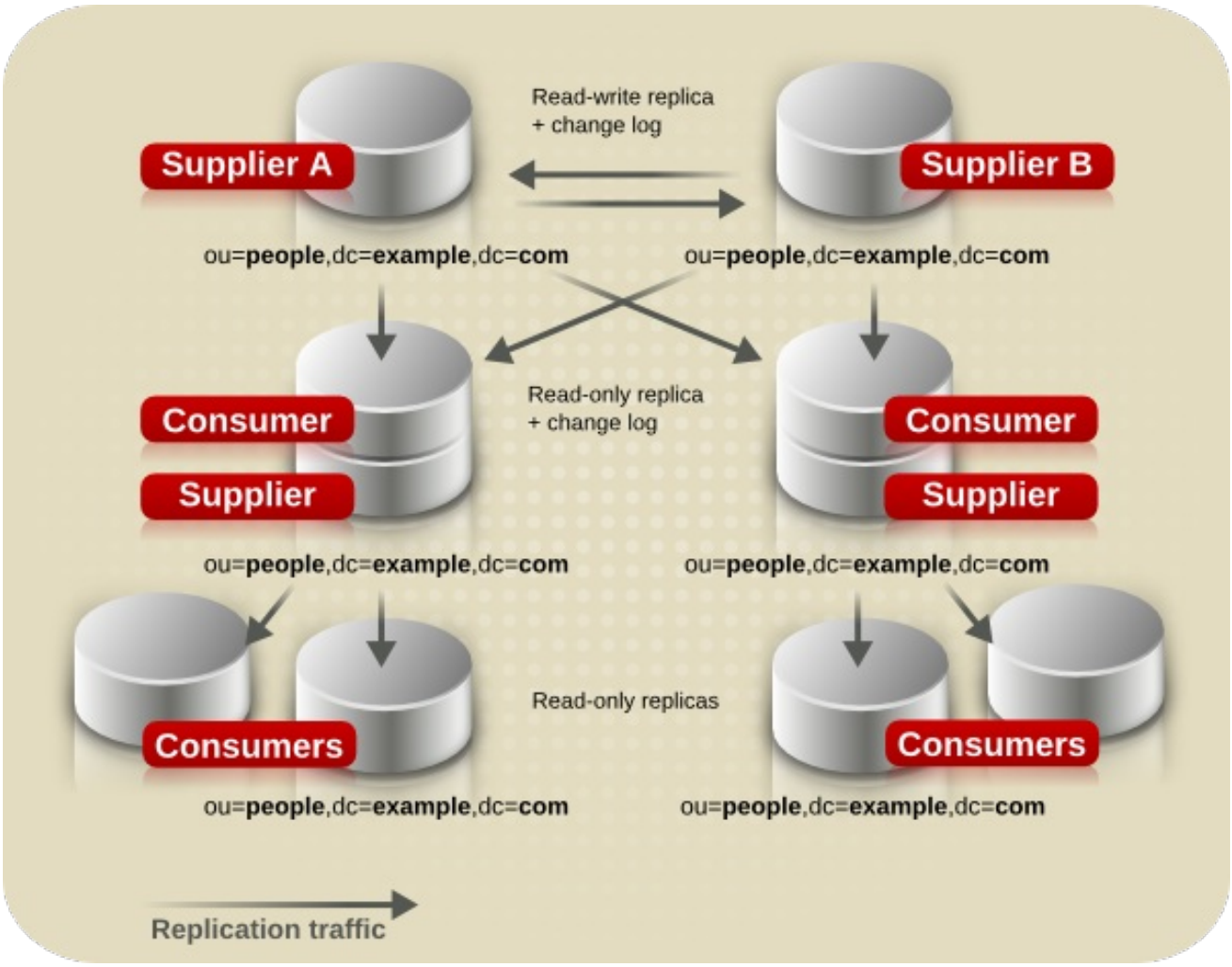
图 7.7. 复制流量和更改日志



7.2.4. 混合环境

可以合并任何复制场景，以满足网络和目录环境的需求。种常见组合是使用具有级联配置的多层次配置。

图 7.8. 合并多路和捕获复制



7.3. 定义复制策略

复制策略由必须提供的服务决定。要确定复制策略，首先对网络、用户、应用程序及其如何使用目录服务的调查开始。

- **评估网络、流量负载和资源对目录服务的资源要求。**

  请参阅 第 7.3.1 节 “执行复制调查”、第 7.3.3 节 “复制资源要求” 和 第 7.3.4 节 “管理多容量复制所需的磁盘空间”。

- **如果不同位置或部分不同的用户有多个用户，或者一些服务器不安全，则使用部分 复制 来排除敏感或 seldom-modified 信息，以在不损害敏感信息的情况下维持数据完整性。**

  如需更多信息，请参阅 第 7.3.2 节 “使用 Fractional Replication 复制复制所选属性”。

- **如果网络在地理区域扩展，则多个站点有多个目录服务器，本地数据供应商通过多层次复制连接了本地数据供应商。**

  如需更多信息，请参阅 第 7.3.5 节 “跨 Wide-Area 网络复制”。

- **如果高可用性是主要关注的，请在单一站点上创建一个包含多个目录服务器的数据中心。单 supplier 复制提供了读取失败，而 multi-supplier 复制则提供了 write-failover。**

  如需更多信息，请参阅 第 7.3.6 节 “使用复制进行高可用性”。

- 如果本地可用性是主要关注，请使用复制将数据分布到位于全球本地办事处的目录服务器。所有信息的一个主要副本可以在单一位置（如公司总部）维护，或者每个本地站点都可以管理与其相关的 DIT 部分。

  如需更多信息，请参阅 第 7.3.7 节 “使用 Replication 进行本地可用性”。

- 在所有情况下，平衡由目录服务器提供服务的请求负载，并避免网络拥塞。

  如需更多信息，请参阅 第 7.3.8 节 “使用 Replication 进行负载均衡”。

在规划复制策略后，可以部署目录服务。最好将目录服务部署到阶段，因为这可让管理员根据企业在目录服务上的负载来调整目录服务。除非负载分析基于已在运行的目录，准备更改目录服务，因为目录的现实需求变得很明显。

7.3.1. 执行复制调查

在站点问卷调查中收集有关网络质量和使用情况的信息，以帮助定义复制策略：

- LAN 和 WAN 的质量可以连接不同的构建或远程站点以及可用带宽的数量。
- 用户的物理位置、每个站点中的用户数量及其使用模式；这就是他们打算如何使用目录服务。
- 访问目录服务的应用程序数量以及读取、搜索的相对百分比，以及比较操作与写入操作。
- 如果消息传递服务器使用 目录，请找出它处理的每个电子邮件消息所执行的操作数量。依赖于目录服务的其他产品通常是身份验证应用程序或子目录应用程序等产品。对于每个，确定目录服务中执行的操作的类型和频率。
- 存储在目录服务中的条目的数量和大小。

管理人类资源数据库或财务信息的站点可能比那些只用于电话目的的工程人员在目录服务上重取负载。

7.3.2. 使用 Fractional Replication 复制复制所选属性

部分复制允许管理员选择从供应商传输到消费者（或其他供应商）的一系列属性。因此，管理员可以在不复制包含的所有信息的情况下复制数据库。

每个复制协议都启用和配置部分复制。属性排除所有条目同样同样地应用。至于消费者服务器方面，排除的属性始终没有值。因此，客户端对消费者服务器执行搜索永远不会看到排除的属性。类似地，应当执行搜索指定其过滤器中的这些属性，不会匹配条目。

在以下情况下，部分复制特别有用：

- 如果消费者服务器使用较慢的网络进行连接，但不经常更改的属性或更大属性，如 jpegPhoto 会导致网络流量减少。
- 如果使用者服务器放置在不受信任的网络中，例如公共互联网，不包括敏感属性，如电话数字，即使服务器的访问控制措施被破坏或机器被攻击者破坏，也不再提供任何级别的保护。

在第 8 章" 管理指南 "中的复制协议和供应商配置小节中介绍了配置部分复制。

7.3.3. 复制资源要求

使用复制需要更多资源。在定义复制策略时请考虑以下资源要求：

- *磁盘用量 - 在供应商服务器中，更改日志在每个更新操作后写入。接收许多更新操作的供应商服务器可能会遇到更多磁盘用量。*

注意

每个供应商服务器都使用单一的 changelog。如果供应商包含多个复制的数据库，则更改日志会更频繁使用，磁盘使用量越高。

- *服务器线程 - 每个复制协议使用一个服务器线程。因此，客户端应用程序可用的线程数量会减少，可能会影响客户端应用程序的服务器性能。*
- *文件描述符 - 服务器可用的文件描述符数量会减少更改日志（一个文件描述符）和每个复制协议（每个协议一个文件描述符）。*

7.3.4. 管理多容量复制所需的磁盘空间

multi-supplier 副本维护额外的日志，包括目录编辑的更改日志、更新条目的状态信息以及删除条目的 tombstone 条目。执行多层次复制需要此信息。由于这些日志文件会变得非常大，因此需要定期清理这些文件，以便防止浪费磁盘空间。

有四个属性可以为 multi-supplier 副本配置 changelog maintenance。两个位于 cn=changelog5 下，直接与修剪更改日志相关：

- ***nsslapd-changelogmaxage** 设置更改日志中条目可以达到的最长期限；一旦一个条目比这个限制旧，它会被删除。这样会使更改日志无限期地增大。*
- ***nsslapd-changelogmaxentries** 设置 changelog 中允许的最大条目数。与**nsslapd-changelogmaxage** 一样，这也会修剪更改日志，但要小心设置。这必须足够大，以允许一组完整的目录信息或多层次复制可能无法正常工作。*

*其他两个属性位于 **cn=replica, cn=**suffixDN, **cn=mapping tree, cn=config** 中的复制协议条目下。这两个属性与在 changelog 中保留的维护信息相关，即 tombstone 和 state 信息，而不是目录编辑信息。*

- ***nsDS5ReplicaPurgeDelay** 设置 tombstone （删除）条目和状态信息的最大期限。一旦 tombstone 或 state information 条目早于这个年龄，就可以删除它。这与 **nsslapd-changelogmaxage** 属性不同，其中 **nsDS5ReplicaPurgeDelay** 值仅适用于 tombstone 和状态信息条目；**nsslapd-changelogmaxage** 适用于 changelog 中的每个条目，包括目录修改。*
- ***nsDS5ReplicaTombstonePurgeInterval** 设置服务器运行清除操作的频率。在这个间隔里，Directory 服务器会运行内部操作来清理 changelog 中的 tombstone 和 state 条目。确保最长期限超过复制更新调度的最长期限，或者多层次复制可能无法正确更新副本。*

管理复制和更改日志的参数在第 2 章"核心配置属性"中所述，如 配置、命令和文件参考 中所述。

7.3.5. 跨 Wide-Area 网络复制

广域网通常具有较高的延迟、更高的带宽延迟产品，速度低于局域网。当供应商和消费者使用广域网连接时，目录服务器支持有效的复制功能。

在以前的目录服务器版本中，用于传输供应商和用户之间的条目和更新非常敏感，因为供应商仅发送一个更新操作，然后等待消费者的响应。这会导致减少延迟较高的吞吐量。

供应商在不等待响应的情况下向消费者发送多个更新和条目。因此，在延迟高的网络中，许多复制操作可以在网络上传输，而复制吞吐量则类似于在本地区域网络中实现的。

注意

如果供应商连接到运行比 7.1 之前的版本的 Red Hat Directory Server 版本的另一个供应商，它会回退到旧的复制机制来实现兼容性。因此，需要在供应商和消费者服务器上运行至少版本 7.1，才能实现对延迟的复制。

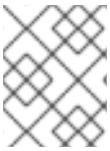
对于目录服务器和网络连接效率，同时存在性能和安全问题：

- 如果跨公共网络（如互联网）执行复制，则强烈建议使用 TLS。这种保护措施可防止停止复制流量。
- 对网络使用 T-1 或更快互联网连接。
- 在创建用于跨域网络复制的协议时，请避免在服务器间持续同步。复制流量可能会消耗大量带宽，并减慢整个网络和互联网连接的速度。
- 初始化消费者时，不会立即初始化消费者；相反，使用文件系统副本初始化速度要快于在线初始化或从文件初始化。有关使用文件系统副本初始化的信息，请参阅 红帽目录服务器管理指南。

7.3.6. 使用复制进行高可用性

使用复制以防止丢失单一服务器导致目录服务不可用。至少将本地目录树复制到至少一个备份服务器。

某些目录架构师认为每个物理位置应复制三次，以获得最大数据可靠性。将复制用于容错的程度取决于环境和个人首选项，但根据目录服务使用的硬件和网络的质量，以此作为这一决定。不可靠硬件需要更多备份服务器。



注意

不要使用复制作为常规数据备份策略的替代。有关备份目录数据的详情，请查看 红帽目录服务器管理指南。

要保证所有目录客户端的写入故障切换，请使用多层次复制场景。如果 read-failover 足够了，请使用单层复制。

LDAP 客户端应用程序通常只能配置为只搜索一个 LDAP 服务器。除非有自定义客户端应用程序通过位于不同 DNS 主机名的 LDAP 服务器轮转，否则 LDAP 客户端应用程序只能配置为查找目录服务器的单一 DNS 主机名。因此，可能需要使用 DNS 轮循或网络排序来向备份目录服务器提供故障转移。有关设置和使用 DNS 循环或网络排序的详情，请参考 DNS 文档。

7.3.7. 使用 Replication 进行本地可用性

复制本地可用性的需求由网络的质量以及站点的活动决定。另外，请仔细考虑目录服务中包含的数据的性质，如果数据暂时不可用，则会给企业带来后果。对数据进行更关键任务，对系统的容错程度越低，导致网络连接不佳。

将复制用于本地可用性，理由如下：

- 保留数据的本地主副本。

  对于大型的跨国企业来说，这是一个重要的策略，需要仅维护特定国家或地区员工感兴趣的目录信息。拥有本地主副本对于任何企业来说也很重要，因为任何位于部门或机构级别控制数据的任何企业来说也很重要。

- 缓解不可靠或间歇性可用的网络连接。

如果出现不可靠的 WAN，则可能会出现间歇性网络连接，如国际网络中发生。

- 要偏移定期，极高的网络负载可能会导致目录服务的性能被严重降低。

  在具有陈旧网络的企业中，性能可能也会受到影响，在正常的工作时间内可能会遇到这些状况。

7.3.8. 使用 Replication 进行负载均衡

复制可以通过多种方式平衡目录服务器上的负载：

- 通过将用户的搜索活动分散在多个服务器中。
- 将服务器指定给只读活动（只在供应商服务器上发生）。
- 通过将特殊服务器专用于特定的任务，如支持邮件服务器活动。

平衡网络的工作负载是由目录数据复制执行的重要功能。尽可能地将数据移动到可以使用合理快速、可靠的网络连接访问的服务器。最重要的注意事项是服务器与目录用户之间的网络连接速度和可靠性。

目录条目通常平均大约为 1 kilobyte(KB)。因此，每个目录查找都会在网络负载中添加一个 KB。如果目录用户每天执行十个目录查找，那么对于每个目录用户来说，每个目录用户都会增加网络负载，每天大约有 10KB 的网络负载。如果站点有一个缓慢、大量加载或不可靠的 WAN，那么请考虑将目录树复制到本地服务器。

另外，请考虑，本地可用数据的好处是可能需要因为复制导致的网络负载增加而造成的。如果整个目录树复制到远程站点，例如，这可能会给网络增加大量压力，这与用户的目录查找造成的流量进行比较。当目录树经常改变时，这尤其如此，但在远程站点中只有一些用户每天执行一些目录查找。

表 7.1 “网络上的复制和远程查找的影响” 比较复制目录 100 万条目的成本（即每天更改这些条目 10% 的数据），同时需要花费少量远程站点，每天执行 10 个查找。每个情况下，目录条目的平均大小被认为是 1KB。

表 7.1. 网络上的复制和远程查找的影响

加载类型	对象（object）[a]	access/Day[b]	avg.条目大小	Load
复制	100万次	100,000	1KB	100Mb/day
远程查找	100	1,000	1KB	1Mb/day

[a] 对于复制，对象 指的是数据库中的条目数量。对于远程查找，它指的是访问数据库的用户数量。

[b] 对于复制，Accesses/Day 基于 10% 对需要复制的数据库的变化率。对于远程查找，它基于每日的远程用户的查询。

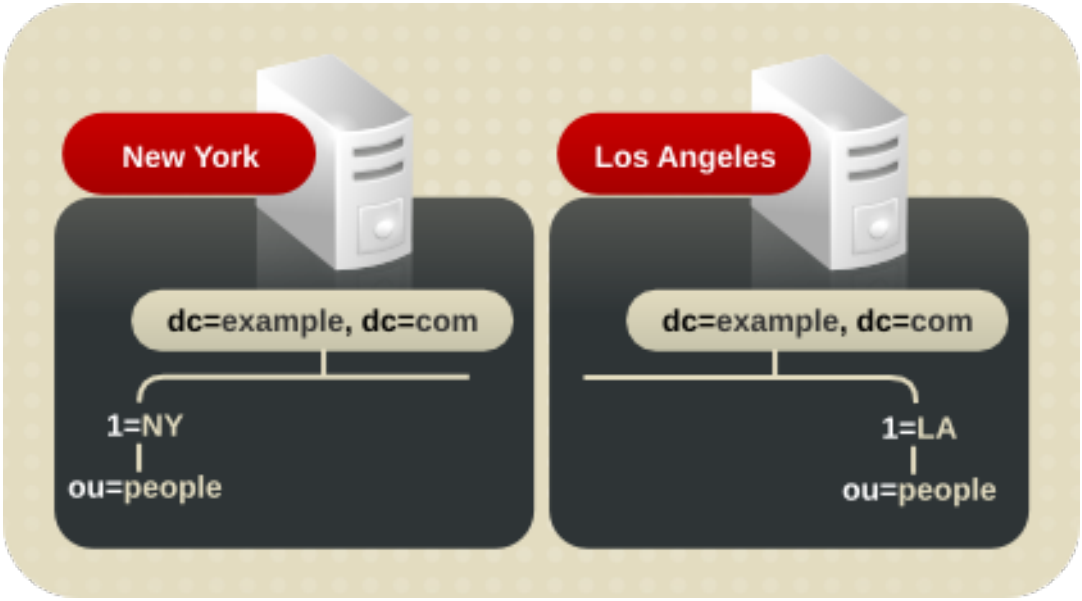
由于复制与普通目录使用情况导致的加载差异不同，因此使用复制进行网络负载平衡需要。另一方面，本地可用目录数据的好处可能远超过网络负载的注意事项。

在为本地站点和网络过载和使用调度复制时，最好发生数据。有关数据一致性和复制计划的更多信息，请参阅 第 7.1.2 节 “数据一致性”。

7.3.8.1. 网络负载均衡示例

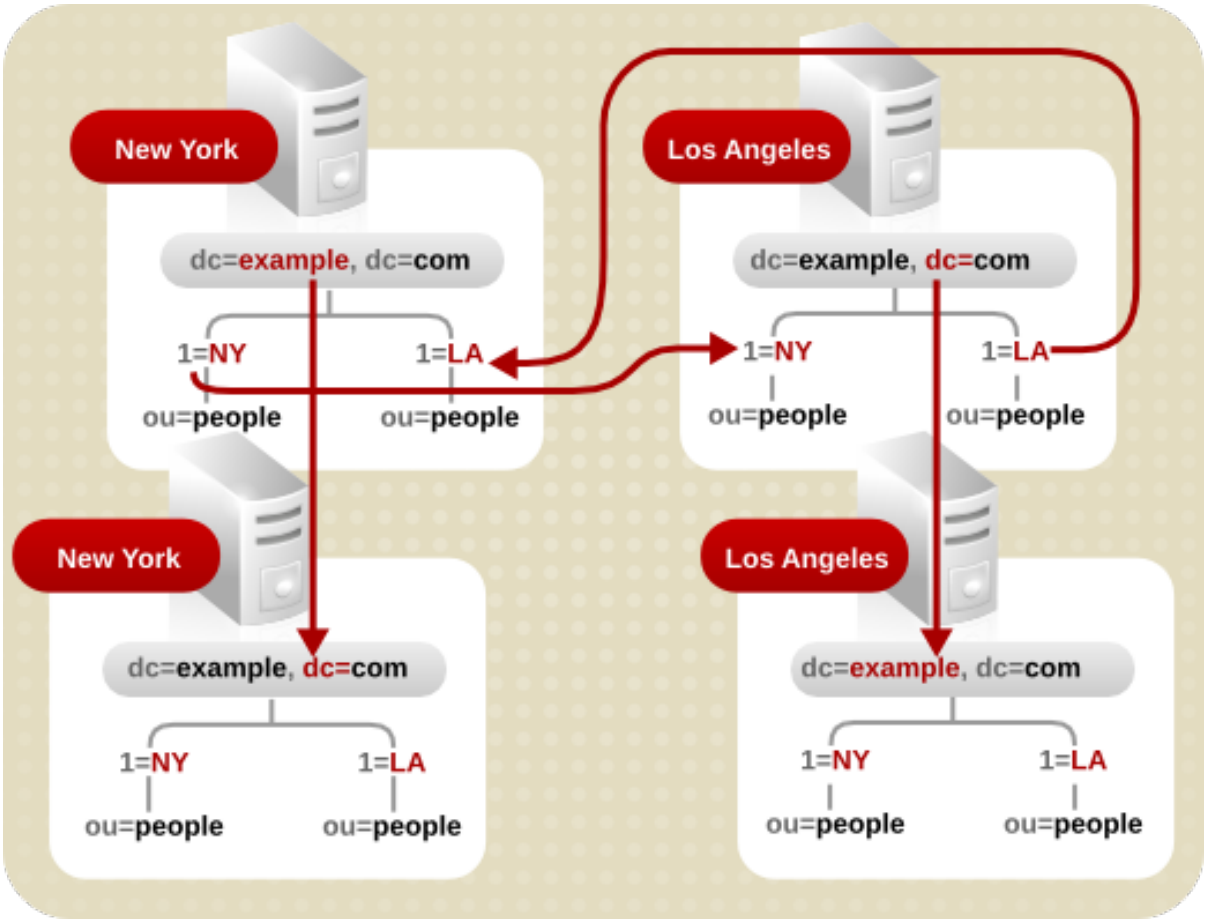
在本示例中，企业在纽约和 Los Ans Angeles 处设有办事处，每个办事处都有其管理的特定子树。

图 7.9. 在远程办公室管理企业子树



每个办公室包含一个高速网络，但两个城市之间的连接不可靠。平衡网络负载：

1. *为每个办公室选择一个服务器作为本地管理数据的供应商服务器。*
2. *将本地管理的数据从该服务器复制到远程办公室中对应的供应商服务器。*
3. *将每个供应商服务器上的目录树（包括从远程办公室提供的数据）复制到至少一个本地目录服务器，以确保目录数据的可用性。对本地管理的后缀使用多supplier 复制，为接收远程服务器数据的主副本的后缀进行级联复制。*



7.3.8.2. 用于提高性能的负载平衡示例

假设企业有以下特征：

- 使用一个目录服务器，其中包含支持 100 万用户的 150 万个条目
- 每个用户每天执行十个目录查找
- 使用一个消息传递服务器，每天处理 2,500 万邮件
- 消息传递服务器针对它处理的每个邮件执行五个目录查找

这相当于每天进行 1 千万次的用户查找，每天 1.25 亿次的电子邮件查找，总计每天 1.35 次的目录查找。

随着营业日的 8 小时工作日和用户分布到四个时区，例如，在四个时间段内的工作日（或峰值使用）将延长至 12 小时。因此，该服务必须在 12 小时时间内支持 1.35 亿目录查找。此等于为每秒 3,125 查找（135,000,000 /(60*60*12)）。

表 7.2. 计算目录服务器负载

访问类型	类型数	每日访问	总访问数
用户查找	100 万次	10	1,000 万
电子邮件查找	2,500 万	5	125 million
组合访问			135 million
总计		1.35 亿(3,125/second)	

如果运行 Directory 服务器的硬件支持每秒读取 500 个，则必须使用至少 6 个或 7 个目录服务器来支持此负载。对于拥有一百万目录用户的企业，为本地可用性添加更多目录服务器。

复制方法有几种：

- 在一个城市中配置两个目录服务器以处理所有写入流量。

  此配置假定应该对所有目录数据有单一的控制点。

- 使用这些供应商服务器复制到一个或多个 hub 供应商。

  目录服务服务的读取、搜索和比较应针对于消费者服务器的请求，从而释放供应商服务器来处理写入请求。

- 使用 hub 供应商将复制到整个企业的本地站点。

  复制到本地站点有助于平衡服务器的工作负载和 WAN，以及确保目录数据的高可用性。

- 在每个站点，至少复制一次以确保高可用性，至少用于读取操作。

- 使用 DNS sort 以确保本地用户始终找到他们可以用于目录查找的本地目录服务器。

7.3.8.3. Small 站点的 Replication 策略示例

example Corp. 具有以下特征：

- 整个企业都包含在一个构建中。
- 构建速度很高（每秒100 Mb/秒）和轻量使用的网络。
- 网络非常稳定，服务器硬件和操作系统平台是可靠的。
- 单一服务器可轻松处理站点的负载。

在这种情况下，Example Corp. 决定在主服务器关机以进行维护或硬件升级时，至少复制一次，以确保主服务器关机。另外，设置一个 DNS 轮循，以便在其中一个目录服务器不可用时提高 LDAP 连接性能。

7.3.8.4. 大型站点的复制策略示例

随着 example Corp. 的增长，它保留了以前的特征（如 第 7.3.8.3 节 "Small 站点的 Replication 策略示例" 中），它有一些变化：

- 企业包含在两种独立构建中。
- 构建之间有较慢的连接，这些连接在正常工作时间内非常忙。

随着对网络需要的改变，那么 Corp. 的管理员需要调整其复制策略：

- 在两个构建中选择一个服务器，以包含目录数据的主副本。

  此服务器应放在构建中，其中包含负责目录数据的主副本的最大人员数量。我们应该将这一构建视为构建 A。

- 在构建 A 中至少复制一次以实现高可用性目录数据。

  使用多supplier 复制配置来确保写入失败。

- 在第二个构建(Building B)中创建两个副本。
- 如果供应商和消费者服务器之间不需要关闭一致性，请调度复制，使其仅在非高峰期时间进行。

7.4. 使用带有其他目录服务器功能的复制

复制与其他目录服务器功能交互，以提供高级复制功能。以下小节描述了用于更好地设计复制策略的功能交互。

7.4.1. 复制和访问控制

目录服务将 ACI 存储为条目属性。这意味着 ACI 会与其他目录内容一起复制。这很重要，因为 Directory 服务器在本地评估 ACI。

有关为目录设计访问控制的更多信息，请参阅 第 9 章 设计安全目录。

7.4.2. 复制和目录服务器插件

复制操作适用于由 Directory 服务器提供的大多数插件。使用以下插件进行多层次复制时，有一些例外和限制：

- 属性唯一插件

  Attribute Uniqueness Plug-in validate 属性值已添加至本地条目，以确保所有值都是唯一的。但是，这个检查直接在服务器上完成，而不是从其他供应商复制。例如，Example Corp. 要求 mail

属性是唯一的，但两个用户同时添加相同的 mail 属性到两个不同的供应商服务器。只要没有命名冲突，就没有复制冲突，但 mail 属性不是唯一的。

- 参考完整性插件

  参考完整性可以和多层次复制一同工作，只要该插件仅在多层次集中的一个供应商上启用。这样可确保仅在其中一个供应商服务器上发生引用完整性更新，并传播到其他供应商服务器。

注意

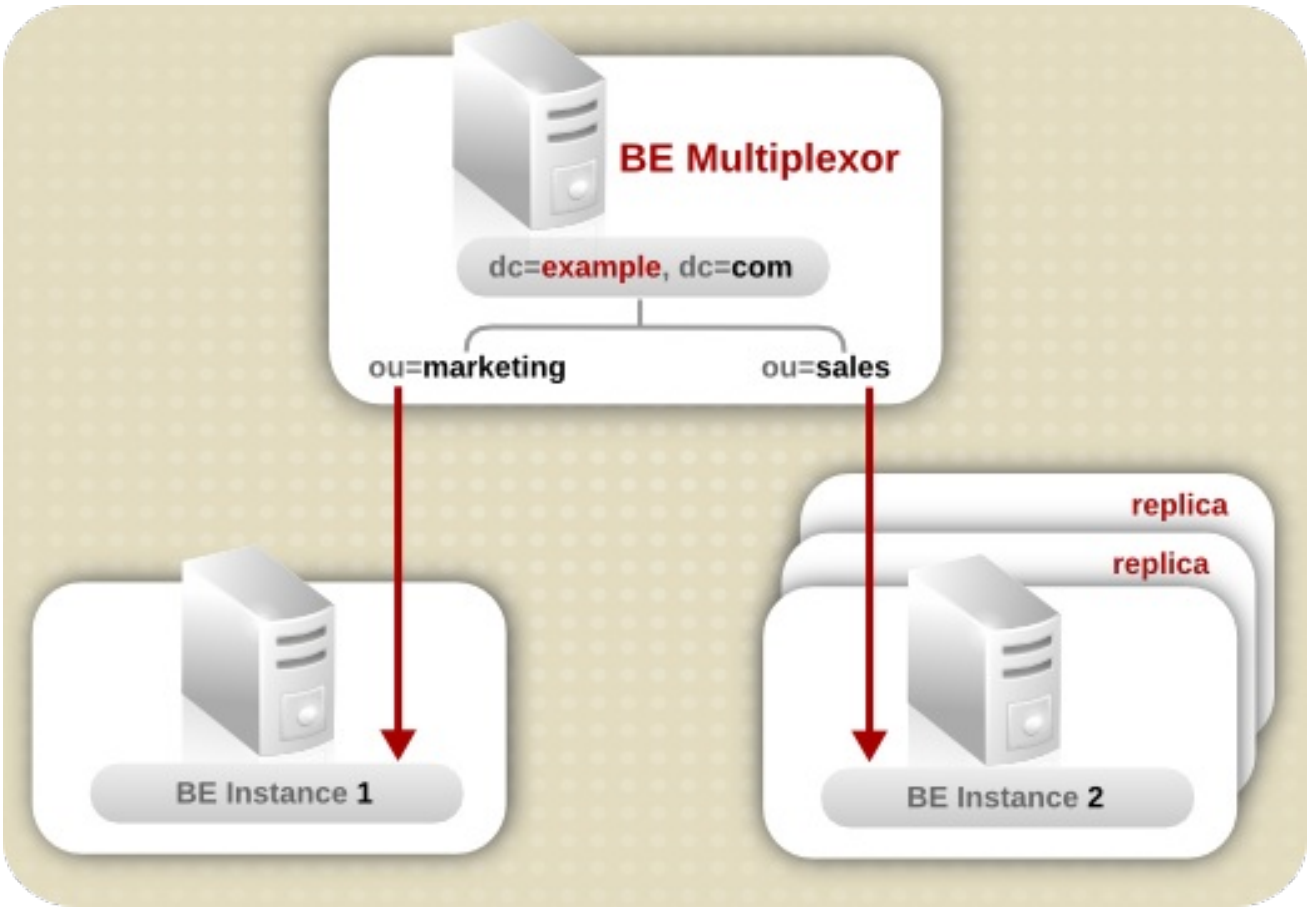
默认情况下，这些插件被禁用，必须手动启用它们。

7.4.3. 复制和数据库链接

通过串联来分发目录条目，包含数据库链接的服务器会引用包含实际数据的远程服务器。在此环境中，无法复制数据库链接本身。但是，可以复制包含远程服务器中实际数据的数据库。

不要使用复制过程作为数据库链接的备份。必须手动备份数据库链接。有关链和条目分发的详情请参考 第 6 章 设计目录拓扑。

图 7.10. 复制链的数据库



7.4.4. 模式复制

对于标准架构，在将数据复制到消费者服务器之前，供应商服务器会检查自己的模式版本是否与消费者服务器中的架构版本同步。适用以下条件：

- 如果供应商和消费者的 schema 条目都相同，复制操作将继续。

- 如果供应商服务器的模式版本比消费者中存储的版本更新，则供应商服务器会将其架构复制到消费者，然后再继续数据复制。
- 如果供应商服务器的模式版本早于使用者上存储的版本，服务器可能会在复制过程中返回许多错误，因为消费者的 schema 无法支持新的数据。

注意

模式复制仍发生，即使供应商和副本之间的模式不匹配。

Replicatable 更改包括通过 Web 控制台进行的模式更改、通过 **ldapmodify** 进行的更改，以及直接对 **99user.ldif** 文件进行的更改。自定义架构文件以及对自定义架构文件所做的任何更改都不会被复制。

消费者可能包含来自两个供应商的复制数据，各自有不同的架构。无论供应商最近一次胜利，其架构都会传播到消费者中。

警告

永不在消费者服务器上更新架构，因为供应商服务器无法解决发生冲突，并且复制失败。架构应该在复制拓扑中的供应商服务器上维护。

相同的目录服务器可存放作为供应商和作为消费者的只读副本的读写副本。因此，请始终识别将充当 schema 的供应商的服务器，然后在本供应商和复制环境中作为 schema 信息的用户在复制环境中设置复制协议。

复制架构不需要特殊复制协议。如果在供应商和消费者之间配置了复制，则默认发生 schema 复制。

有关 schema 设计的详情请参考 第 3 章 设计目录架构。

自定义架构

如果标准 **99user.ldif** 文件用于自定义模式，则这些更改将复制到所有消费者。

自定义架构文件必须复制到每台服务器上，才能保持所有服务器上的相同模式文件中的信息。自定义架构文件以及对这些文件的更改不会复制，即使它们通过 Web 控制台或 **ldapmodify** 进行。

如果有自定义架构文件，请确保在供应商更改时将这些文件复制到所有服务器。复制完所有文件后，重启服务器。

有关自定义模式文件的详情，请参考 第 3.4.7 节 “创建自定义架构文件”。

7.4.5. 复制和同步

要在 Directory 服务器中传播同步的 Windows 条目，请在多层次环境中使用同步。同步协议应保持在最低程度上，每个部署最好有一个。多层次复制允许 Windows 信息在网络中可用，同时限制对单个目录服务器的数据访问点。

第 8 章 设计同步

在对现有站点(第 2.3 节 "执行站点问卷调查")进行站点调查时需要考虑的一个重要因素是包括 Active Directory 目录服务的结构和数据类型。通过 Windows 同步，可以同步现有 Windows 目录服务并将其与 Directory 服务器集成，包括创建、修改和删除目录服务器上的 Windows 帐户，或者相反的 Windows 上的 Directory Server 帐户。这提供了在目录服务中维护目录信息完整性的有效方法。

8.1. WINDOWS 同步概述

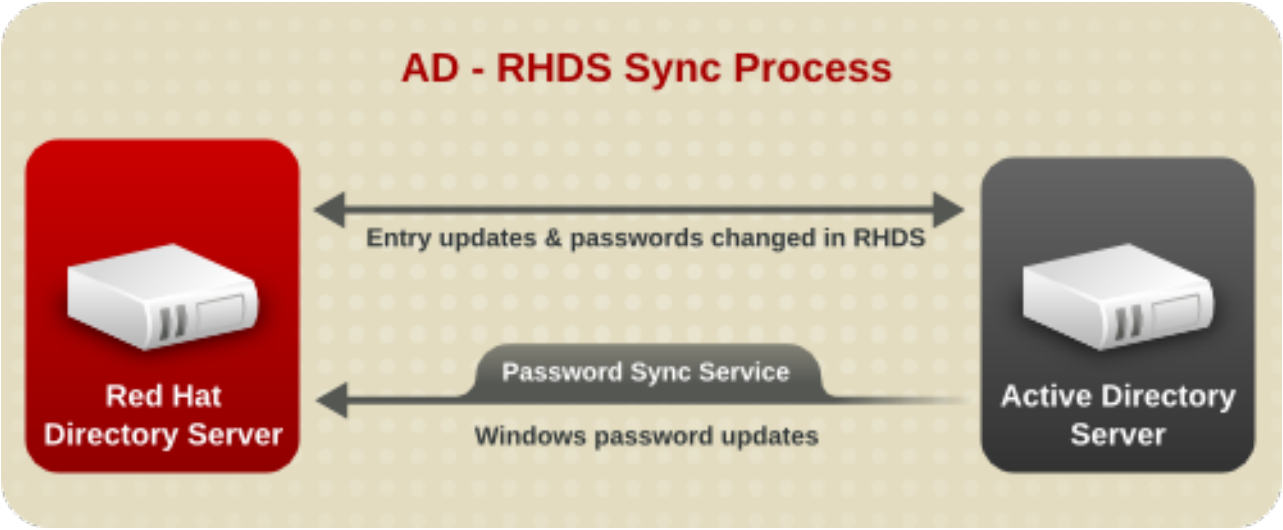
同步过程类似于复制过程：该插件启用和发起，并且通过同步协议启动，并且目录更改的记录会被维护，并根据该日志发送更新。

完整 Windows 同步过程有两个部分：

- 用户和组同步.与多倍复制一样，用户和组条目将通过插件同步，默认情况下是启用的。与用于多层次复制的变更记录也用于将更新从 Directory 服务器发送到 Windows 同步对等服务器作为 LDAP 操作。服务器还针对其 Windows 服务器执行 LDAP 搜索操作，将 Windows 条目所做的更改同步到对应的目录服务器条目。

- 密码同步.此应用程序捕获 Windows 用户的密码更改，并通过 LDAPS 将这些更改转发回目录服务器。它必须安装在 Active Directory 机器上。

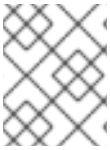
图 8.1. 同步过程



8.1.1. 同步协议

同步由一个或多个同步协议配置和控制。它们与复制协议类似，包含类似的信息，包括 Windows 服务器的主机名和端口号，以及要同步的子树。Directory 服务器通过 TLS 使用 LDAP 或 LDAP 连接到发送和接收更新，连接到其对等 Windows 服务器。

单个 Windows 子树与单个 Directory Server 子树同步，反之亦然。与连接数据库的复制不同，同步是在后缀、部分目录树结构间进行。因此，在设计目录树时，请考虑应当与 Directory 服务器同步的 Windows 子树，以及设计或添加对应的 Directory Server 子树。同步的 Windows 和目录服务器后缀都在同步协议中指定。对应子树中的所有条目都可用于同步，包括不是指定后缀直接子级的条目。



注意

任何子代容器条目需要由管理员在 Windows 服务器上单独创建；Windows Sync 不会创建容器条目。

8.1.2. changelogs

Directory Server 维护一个 changelog，这是记录发生的修改的数据库。Windows Sync 使用 changelog 来协调并发送对 Windows 同步对等服务器所做的更改。使用 Active Directory 的 Dirsync 搜索功能可找到对 Windows 服务器中的条目的更改。因为 Active Directory 端没有 changelog，所以默认会发布 Dirsync 搜索。使用 Dirsync 可确保仅检索自上次搜索以来更改的条目。

8.2. 支持的 ACTIVE DIRECTORY 版本

Windows 2008 R2 和 Windows 2012 R2 在 32 位和 64 位平台上均支持 Windows 同步和密码同步服务。

8.3. 规划 WINDOWS 同步

在设置同步前评估信息、Windows 服务器和其他注意事项的信息、Windows 服务器和其他注意事项可能很有用，类似于组织数据或规划复制的站点调查。

8.3.1. 资源要求

同步使用服务器资源。在定义复制策略时请考虑以下资源要求：

- 磁盘用量 - 更改日志在每个更新操作后写入。接收许多更新操作的服务器可能会看到更多磁盘用量。另外，为所有复制数据库和同步数据库维护单一的 changelog。如果供应商包含多个复制和同步的数据库，则更改日志会更频繁使用，磁盘使用量越高。
- 服务器线程 - 同步协议使用一个服务器线程。
- 文件描述符 - 服务器可用的文件描述符数量会减少更改日志（一个文件描述符）和每个复制与同步协议（每个协议一个文件描述符）。
- LAN 和 WAN 质量可以连接不同的构建或远程站点，以及可用带宽的数量。
- 目录中存储条目的数量和大小。

与站点相比，管理人员资源数据库或财务信息的网站可能包含使用目录进行简单的电话书的工程人员，该站点可能会对目录进行负担。

8.3.2. 为 Changelog 管理磁盘空间

与多器复制一样，同步需要更改日志来跟踪更新条目的状态信息的目录编辑和日志条目，以及删除条目的 tombstone 条目。此信息是同步所必需的。由于这些日志文件会变得非常大，因此需要定期清理这些文件，以便防止浪费磁盘空间。

维护 changelog 的四个属性：两个位于 **cn=changelog5** 下，直接与修剪更改日志相关：

- **nsslapd-changelogmaxage** 设置更改日志中条目可以达到的最长期限；一旦一个条目比这个限制旧，它会被删除。这样会使更改日志无限期地增大。
- **nsslapd-changelogmaxentries** 设置 changelog 中允许的最大条目数。与 **nsslapd-changelogmaxage** 一样，这也会修剪更改日志，但要小心设置。这必须足够大，以便完整的目录信息或同步可能无法正常工作。

其他两个属性位于 **cn=**sync_agreement,**cn=WindowsReplica,cn=**suffixDN,**cn=mapping tree,cn=config** 中的同步协议条目下。这两个属性与在 changelog 中保留的维护信息相关，即 tombstone 和 state 信息，而不是目录编辑信息。

- **nsDS5ReplicaPurgeDelay** 设置 tombstone （删除）条目和状态信息的最大期限。一旦 tombstone 或 state information 条目早于这个年龄，就可以删除它。这与 **nsslapd-changelogmaxage** 属性不同，其中 **nsDS5ReplicaPurgeDelay** 值仅适用于 tombstone 和状态信息条目； **nsslapd-changelogmaxage** 适用于 changelog 中的每个条目，包括目录修改。

- **nsDS5ReplicaTombstonePurgeInterval** 设置服务器运行清除操作的频率。在这个间隔里，Directory 服务器会运行内部操作来清理 changelog 中的 tombstone 和 state 条目。确保最长期限超过复制更新调度的最长期限，或者多层次复制可能无法正确更新副本。

管理复制和更改日志的参数在第 2 章"核心配置属性"中所述，如 配置、命令和文件参考 中所述。

8.3.3. 定义连接类型

可使用标准端口、使用 TLS/TLS 或启动 TLS（标准端口上的安全连接）进行同步。

虽然不需要，但强烈建议使用 TLS 或其他安全连接进行同步。如果要从 Windows 服务器同步密码，那么必须在两个服务器上启用 TLS，以便同步在安全端口上继续进行。

8.3.4. 考虑数据供应商

数据供应商 是供应商数据源的服务器；这是数据的主要或权威来源。

Windows 和 Directory Server 服务是通过同步协议持续同步的，这样可最小化这两个服务间的潜在的冲突。但是，如果 Directory 服务器是复制部署的一部分，则目录服务器复制场景中的更改和 Windows 域之间可能会发生冲突，具体取决于复制计划。

当数据位于两个不同的目录服务中时，请考虑哪个服务器将成为数据供应商，并确定信息要共享多少。最好的课程是选择一个目录服务来管理数据，并允许同步进程在其他服务上添加、更新或删除条目。

选择一个区域（Windows 域或目录服务器）来管理数据。或者，选择单个目录服务器作为数据供应商并将其与每个 Windows 域同步。如果目录服务器涉及复制，设计复制结构以避免冲突、丢失数据或覆盖数据。

数据的主副本是如何被维护的，这取决于部署的特定需求。无论数据供应商如何维护，都保持简单且一致。例如，不要尝试管理多个站点中的数据，然后在竞争应用程序之间自动交换数据。这样做会导致"上次更改优先"情况，并增加了管理开销。

8.3.5. 确定要同步的子树

只有一个目录服务器子树可以同步到单个 Windows 子树中，建议在目录服务之间仅有一个同步协议。选择或设计用于同步的目录树的部分；考虑设计特别用于同步条目的特殊后缀。

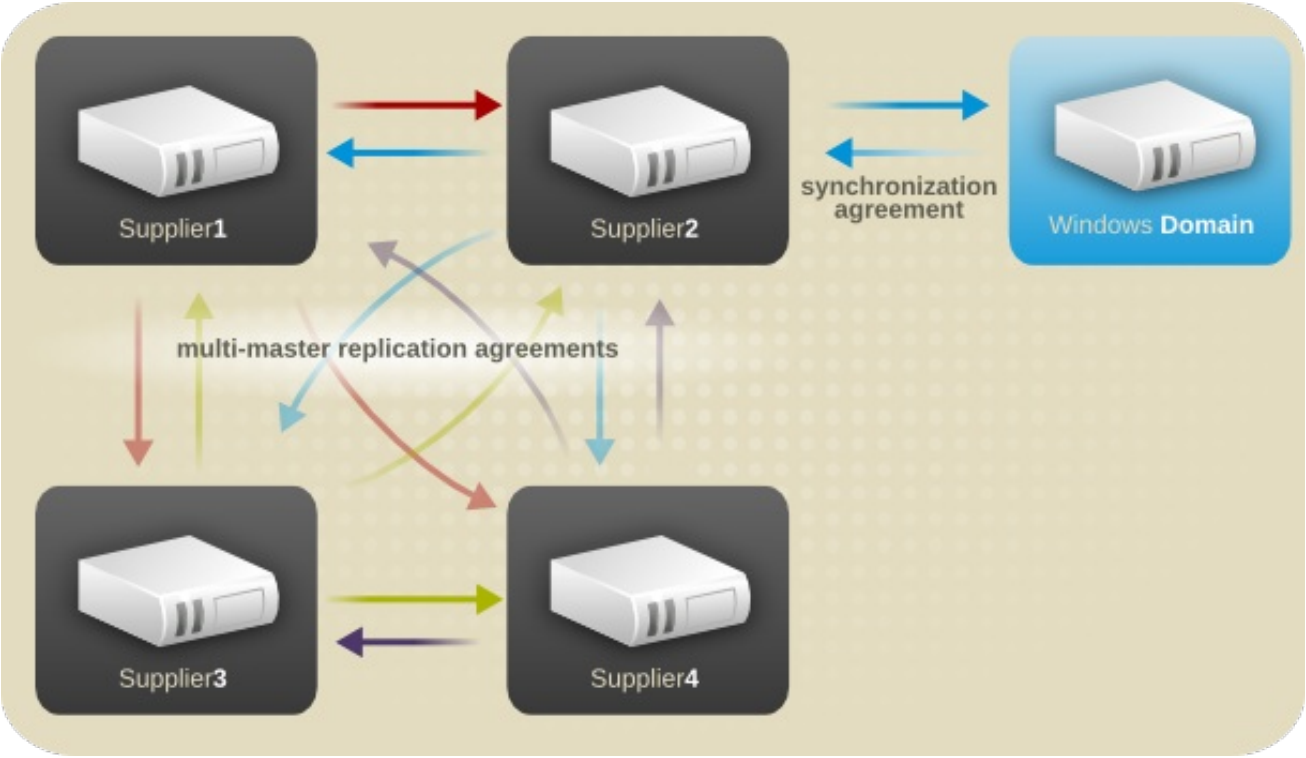
子树计划也应考虑 Active Directory 和 Directory Server 目录之间可能对应的条目，但不属于同步子树的范围。同步进程实际上从根 DN 开始，开始评估条目以进行同步。条目会根据 Active Directory 中的 **samAccount** 和 Directory Server 中的**uid** 属性关联。如果一个条目（基于 **samAccount/uid** 关系）已从同步子树中删除，同步插件请注意，因为它已被删除或移动。这是对条目不再同步的同步插件的信号。该问题是同步过程需要一些配置来确定如何处理该移动条目。同步协议中可以设置三个选项：删除对应的 Directory Server 条目，忽略更改（默认），或者取消同步条目，但保留其他内容。

8.3.6. 与复制环境交互

同步将目录服务器后缀和子树（例如 **ou=People,dc=example,dc=com**）链接到对应的 Windows 域和子树(**cn=Users,dc=test,dc=com**)。每个子树都只能同步到其他子树，以避免命名冲突和更改冲突。

要利用 Windows 同步，请将它与多层次复制中的目录服务器供应商一起使用，并同步到 Windows 域的成员。这会通过两个目录系统传播更改，同时保持信息集中化且易于维护。它还有助于更轻松地管理数据。

图 8.2. 多Supplier 目录服务器 - Windows 域同步



仅创建任何给定 Windows 域的同步协议。要传播在 Directory Server 中与 Windows 服务器同步的更改和信息，与多供应商一起创建同步协议，最好是复制部署的数据供应商。

8.3.7. 控制同步方向

如 图 8.1 “同步过程” 所示，默认同步是双向 的。这意味着 Active Directory 中的更改将发送到 Directory Server，并在 Directory 服务器上的更改发送到 Active Directory。

通过在同步协议中添加 **oneWaySync** 参数，可以创建 uni-directional 同步。此属性定义要发送更改的方向。

要从 Active Directory 服务器发送更改到 Directory 服务器，值为 **fromWindows**。在这种情况下，在常规同步更新间隔中，Directory 服务器联系 Active Directory 服务器，并发送 DirSync 控制来请求更新。但是，目录服务器不会向其发送任何更改或条目。因此，同步更新包含要发送到和更新目录服务器条目的 Active Directory 更改。

要将更改从 Directory 服务器同步到 Active Directory 服务器，值为**toWindows**。Directory 服务器在正常更新中向 Active Directory 服务器发送条目修改，但它不包括 DirSync 控制，以便它不会从 Active Directory 端请求任何更新。

启用双向同步不会自动防止非同步服务器上的更改，这会导致同步更新之间的同步间不一致。例如，单向同步配置为从 Active Directory 转到 Directory 服务器，因此 Active Directory 为（本质上）数据供应商。如果在 Directory 服务器上修改甚至删除了条目，则目录服务器信息会有所不同，且这些更改永远不会被发送到 Active Directory。在下一次同步更新过程中，Directory 服务器上会覆盖编辑，删除的条目会被重新添加。

要防止数据不一致，请使用访问控制规则来防止在未同步的服务器中编辑或删除同步子树中的条目。目录服务器的访问控制包括在 第 9.7 节 “设计访问控制” 中。对于 Active Directory，请查看适当的 Windows 文档。

8.3.8. 控制要同步的尝试

Windows Sync 提供一些控制，可以控制要同步哪些条目，以获得足够的灵活性来支持不同的部署场景。此控制是通过目录服务器中设置的不同配置属性的设置：

- 在 Windows 子树中，只能将用户和组群条目同步到 Directory Server。在创建同步协议时，可以选择在创建新 Windows 用户和组条目时同步它们。如果在 上 将这些属性设置为，则现有 Windows 条目将同步到目录服务器，并在 Windows 服务器中创建的条目与 Directory Server 同步。
- 与 Active Directory 条目一样，只能同步目录服务器中的用户和组群条目。同步的条目必须具有 **ntUser** 或 **ntGroup** 对象类和所需属性；忽略所有其他条目。

目录服务器密码与其他条目属性同步，因为纯文本密码保留在 Directory Server changelog 中。需要密码同步服务才能捕获 Windows 服务器上所做的更改。如果没有密码同步服务，就无法同步 Windows 密码，因为密码在 Windows 服务器中被哈希化，而且 Windows 哈希功能与 Directory Server 使用的结果不兼容。

8.3.9. 确定要同步的目录数据

Windows Sync 在目录服务之间同步用户和组条目。在决定要同步的子树后，规划要在这些子树中存储的信息，如下所示：

- 目录用户和员工的联系信息，如电话号码、家和办公室地址以及电子邮件地址。
- 交易合作伙伴、客户以及客户的联系信息。
- 用户软件首选项或软件配置信息。
- 群组信息和组成员资格。

  只有组成员处于同步后缀中时，才会同步。不在协议范围内的组成员在两端都保持不变；也就是说，它们被列为适当目录服务上的组的成员，而是组条目中的 **member** 属性与同步对等点同步。

同步条目在同步协议中设置。用户条目与组条目分开同步。另外，删除条目会被单独配置，需要特别地同步删除。

在 Directory Server 中，只有包含 **ntGroup** 或 **ntUser** 对象类和所需属性的条目才会同步；确定应该与 Windows 服务器同步的现有和将来的条目。

确定 目录中应该存在哪些条目后，确定目录中需要维护这些对象的属性。只有 Directory Server 或 Active Directory 的可能属性子集才会被同步。此外，通过同步协议（协调同步）排除某些属性，可以限制这部分属性子集。

根据可用的同步属性，规划这些条目中包含的条目和数据。同步的属性以及 Directory 服务器和 Active Directory 模式之间的区别，请参考 第 8.4 节 “Active Directory 和 Directory Server 间同步的元素”。

8.3.10. 为用户和组群同步 POSIX 属性

在 Active Directory 和 Red Hat Directory Server 之间，所有可能的用户和属性子集都同步。一些属性会被映射，在 Active Directory 和 Directory Server 模式之间有区别，一些属性会被直接匹配。默认情况下，只有这些属性会被同步。

该同步列表中缺少的一个属性是任何与 POSIX 相关的属性。在 Linux 系统上，系统用户和组标识为 POSIX 条目，LDAP POSIX 属性则包含该所需信息。但是，当 Windows 用户同步时，它们具有 **ntUser** 和 **ntGroup** 属性，它们会自动添加将其识别为 Windows 帐户，但没有通过同步 POSIX 属性（即使它们存

在于 Active Directory 条目中），并且不会在 Directory Server 端添加 POSIX 属性。

Posix Winsync API 插件在 Active Directory 和 Directory Server 条目之间同步 POSIX 属性。此插件默认为禁用，但启用后，它可允许识别数据供应商中设置的 POSIX 属性，然后通过与对等服务器同步。如果 Active Directory 用作用户帐户存储，这很有用，因为 POSIX 属性可以直接在 Active Directory 条目中设置，然后在 Directory Server 目录中同步和保留。

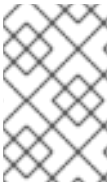
注意

如果启用了插件，则所有 POSIX 属性（如 **uidNumber**、**gidNumber** 和 **homeDirectory**）在 Active Directory 和 Directory Server 条目之间同步。但是，如果新 POSIX 条目或 POSIX 属性添加到 Directory 服务器中的现有条目中，只有 POSIX 属性才会同步到 Active Directory 对应的条目 中。POSIX 对象类(**posixAccount** 用于用户，**posixGroup** 用于组)不会添加到 Active Directory 条目。

8.3.11. 同步密码和安装密码服务

虽然 DirSync 插件默认安装在 Directory Server 并默认启用时，必须在 Windows 机器上安装额外的 Windows 服务 Password Sync 来同步密码。需要该服务将 Windows 服务器上的任何密码更改传送到 Directory Server。

除非安装了 Password Sync 服务，否则不会启用密码同步（同步 **userPassword** 属性）。这意味着，即使 Directory Server 用户条目与 Windows 服务器同步，用户条目在 Windows 域（在其它方面）不会激活，这些同步用户也无法登录到域，因为它们没有密码。



注意

有一个名为 **passwordTrackUpdateTime** 的密码策略属性，它为用户密码最后一次更新记录一个单独的时间戳。这样可以更轻松地同步 Active Directory 和 Directory Server 或其他客户端之间的密码更改。

8.3.12. 定义更新策略

在下一次更新前，不会同步现有目录服务器条目，以包含必要的同步属性。对已同步的 Windows 条目和 Directory Server 条目的修改会在下一次增量更新中执行。作为此策略的一部分，尝试在单个位置管理数据，限制可以更改数据的应用程序，以及调度所需的总体更新（更新不会覆盖或删除现有信息；它们添加新的条目和发送修改）。

默认情况下，Windows 和 Directory 服务器实例将持续同步，并且每五分钟发布一次更改。可以通过手动设置同步协议属性来更改更新间隔(**winSyncInterval**)或设置不同的更新调度(**nsDS5ReplicaUpdateSchedule**)来更改此调度。

8.3.13. 编辑同步协议

通过 Web 控制台配置的基本同步协议设置有关同步的简单信息，如主机和端口信息、同步子树和连接类型。

但是，许多配置可用于多supplier 复制，如部分复制和同步计划，供 Windows-Directory 服务器同步使用。这些设置必须直接添加到同步协议中。

《管理指南》 中介绍了更改默认同步协议，可用的同步协议属性则列在 配置、命令和文件参考 中。

8.4. ACTIVE DIRECTORY 和 DIRECTORY SERVER 间同步的元素

目录服务器中的所有同步条目（无论是来自 Directory Server 还是 Windows 服务器中）都具有以下特殊的同步属性：

- **ntUniqueId** 包含对应 Windows 条目的**objectGUID** 属性的值。此属性由同步进程设置，不应手动设置或修改。
- 当 Windows 条目同步时，会自动设置 **ntUserDeleteAccount**，但必须为 Directory Server 条目手动设置。如果 **ntUserDeleteAccount** 的值为 **true**，则在删除 Directory Server 条目时会删除对应的 Windows 条目。
- **ntDomainUser** 对应于 Active Directory 条目的**samAccountName** 属性。仅用户条目。
- 为同步的 Windows 组自动设置 **ntGroupType**，但必须在同步前手动在 Directory Server 条目上设置。仅组条目。

预定义的属性列表在 Directory 服务器和 Active Directory 条目之间同步。其中一些属性相同，如 Directory 服务器中的 **givenName** 属性与 Active Directory 中的**givenName** 属性匹配。由于 Active Directory 和红帽目录服务器中定义的模式稍有不同，所以其他属性在 Active Directory 和 Red Hat Directory Server 之间映射；其中大多数是 Directory Server 中特定于 Windows 的属性。

8.4.1. 用户属性同步目录服务器和 Active Directory

只有 Directory 服务器和 Active Directory 属性的子集才会被同步。硬编码属性，其定义与条目同步的方式无关。条目中的任何其他属性（位于 Directory 服务器或 Active Directory 中）均不受同步影响。

Directory 服务器和 Active Directory 中使用的一些属性是相同的。这些通常是 LDAP 标准中定义的属性，这些属性在所有 LDAP 服务中很常见。这些属性完全同步到另一个属性。*表 8.2 “Aame in Directory Server 和 Windows Servers 中的用户架构”* 显示 Directory 服务器和 Windows 服务器之间的属性相同。

有些属性会定义相同的信息，但属性的名称或其架构定义有所不同。这些属性在 Active Directory 和 Directory Server 之间映射，这样一个服务器中的 A 属性被视为另一个服务器中的属性 B。对于同步，其中许多属性与 Windows 特定信息相关。*表 8.1 “用户架构在 Directory 服务器和 Active Directory 之间映射”* 显示目录服务器和 Windows 服务器之间映射的属性。

有关 Directory 服务器和 Active Directory 处理某些 schema 元素的方法的更多信息，请参阅 *第 8.4.2 节 “Red Hat Directory Server 和 Active Directory 之间的用户架构差异”*。

表 8.1. 用户架构在 Directory 服务器和 Active Directory 之间映射

目录服务器	Active Directory
cn	name
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon
ntUserLastLogoff	lastLogoff

目录服务器	Active Directory
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

表 8.2. Aame in Directory Server 和 Windows Servers 中的用户架构

cn	physicalDeliveryOfficeName
description	postOfficeBox
destinationIndicator	postalAddress
facsimileTelephoneNumber	postalCode
givenName	registeredAddress
homePhone	sn
homePostalAddress	st
Initials	street
l	telephoneNumber
mail	teletexTerminalIdentifier
manager	telexNumber
mobile	title
o	userCertificate
ou	x121Address

pager	

8.4.2. Red Hat Directory Server 和 Active Directory 之间的用户架构差异

虽然 Active Directory 支持与 Directory Server 相同的基本 X.500 对象类，但管理员应该清楚一些不兼容的问题。

8.4.2.1. cn Attributes 的值

在目录服务器中，**cn** 属性可以是 multi-valued，而 Active Directory 此属性必须只有一个值。当 Directory Server **cn** 属性同步时，只有一个值发送到 Active Directory peer。

这对同步意味着，如果将 **cn** 值添加到 Active Directory 条目，并且该值不是 Directory Server 中**cn** 的值之一，则所有 Directory Server **cn** 值都会用单个 Active Directory 值覆盖。

另一个重要的区别是 Active Directory 使用 **cn** 属性作为其命名属性，其中 Directory 服务器使用 **uid**。这意味着，如果 Directory Server 中编辑 **cn** 属性，则可能完全重命名条目。如果该 **cn** 更改被写入 Active Directory 条目，则该条目将被重命名，并且新命名条目将写回到目录服务器。这只有在 **cn** 属性同步时才会发生。如果没有同步更改，则不会重命名该条目。

8.4.2.2. 密码策略

Active Directory 和 Directory Server 可以强制密码策略，如密码最小长度或最长期限。Windows Sync 不会尝试确保策略一致、强制执行或同步。如果在 Directory 服务器和 Active Directory 中都没有一致的密码策略，那么当同步到其他系统时，在一个系统中进行的密码更改可能会失败。Directory 服务器上的默认密码语法设置模拟 Active Directory 强制执行的默认密码复杂性规则。

8.4.2.3. street 和 streetAddress 的值

Active Directory 将属性 **streetAddress** 用于用户或组的 postal 地址；这是目录服务器使用 **street** 属性的方式。Active Directory 和 Directory 服务器使用 **streetAddress** 和 **street** 属性的方式有两个重要区别：

- 在目录服务器中，**streetAddress** 是 **street** 的别名。Active Directory 也具有 **street** 属性，但它是一个单独的属性，它可以保存独立值，而不是 **streetAddress** 的别名。
- Active Directory 将 **streetAddress** 和 **street** 定义为单值属性，而目录服务器将 **street** 定义为多值属性，如 RFC 4519 中指定的。

由于目录服务器和 Active Directory 处理 **streetAddress** 和 **street** 属性的不同方法，在 Active Directory 和 Directory Server 中设置地址属性时，需要遵循两个规则：

- Windows Sync 将 Windows 条目中的 **streetAddress** 映射到目录服务器中的 **street**。为避免冲突，不应在 Active Directory 中使用 **street** 属性。
- 只有一个目录服务器 **street** 属性值会同步到 Active Directory。如果在 Active Directory 中更改了 **streetAddress** 属性，且 Directory Server 中新值尚不存在，则 Directory Server 中的所有 **street** 属性值都会替换为新的、单一 Active Directory 值。

8.4.2.4. 对初始属性的限制

对于 **initials** 属性，Active Directory 对六个字符实施最大长度约束，但 Directory 服务器没有长度限制。如果向 Directory 服务器添加了一个大于 6 个字符的 **initials** 属性，则该值会在与 Active Directory 条目同步时被修剪。

8.4.3. 在目录服务器和 Active Directory 之间同步组属性

只有 Directory 服务器和 Active Directory 属性的子集才会被同步。硬编码属性，其定义与条目同步的方式无关。条目中的任何其他属性（位于 Directory 服务器或 Active Directory 中）均不受同步影响。

Directory 服务器和 Active Directory 组条目中使用的一些属性是相同的。这些通常是 LDAP 标准中定义的属性，这些属性在所有 LDAP 服务中很常见。这些属性完全同步到另一个属性。表 8.4 “Group Entry 属性是 Directory 服务器和 Active Directory 之间的 Same” 显示 Directory 服务器和 Windows 服务器之间的属性相同。

有些属性会定义相同的信息，但属性的名称或其架构定义有所不同。这些属性在 Active Directory 和 Directory Server 之间映射，这样一个服务器中的 A 属性被视为另一个服务器中的属性 B。对于同步，其中许多属性与 Windows 特定信息相关。表 8.3 “Group Entry 属性映射目录服务器和 Active Directory”显示目录服务器和 Windows 服务器之间映射的属性。

有关 Directory 服务器和 Active Directory 处理某些 schema 元素的方法的更多信息，请参阅 第 8.4.4 节 “Red Hat Directory Server 和 Active Directory 之间的组架构差异”。

表 8.3. Group Entry 属性映射目录服务器和 Active Directory

目录服务器	Active Directory
cn	name
ntGroupAttributes	groupAttributes
ntGroupId	cn name sAMAccountName
ntGroupType	groupType

表 8.4. Group Entry 属性是 Directory 服务器和 Active Directory 之间的 Same

cn	成员
description	ou
l	seeAlso

8.4.4. Red Hat Directory Server 和 Active Directory 之间的组架构差异

虽然 Active Directory 支持与 Directory Server 相同的基本 X.500 对象类，但管理员应该清楚一些不兼容的问题。

支持嵌套组（其中组包含另一个组作为成员），且在 WinSync 中会被同步。但是，Active Directory 对嵌套组的组合实施某些限制。例如，全局组包含域本地组作为成员。目录服务器没有本地和全局组的概念，因此可以在 Directory Server 一侧创建条目，在同步时违反 Active Directory 的限制。

第 9 章 设计安全目录

Red Hat Directory Server 中的数据如何被保护会影响所有之前的设计区域。任何安全设计需要保护目录包含的数据，并满足用户和应用程序的安全性和隐私需求。

本章论述了如何分析安全需求并解释了如何设计目录以满足这些需求。

9.1. 关于安全 THREATS

目录的安全性有很多潜在的威胁。了解最常见的威胁有助于概述整体安全设计。目录安全性的威胁分为三大类别：

- *未授权访问*
- *未授权的篡改*
- *拒绝服务*

9.1.1. 未授权访问

防止目录不受未授权访问的影响可能看似简单，但实施安全解决方案可能比先出现更加复杂。很多潜在的访问点存在于目录信息交付路径上，未授权的客户端可能访问数据。

例如，未授权的客户端可以使用其他客户端的凭据来访问数据。特别是当目录使用未保护的密码时。未授权的客户端也可以在合法客户端和目录服务器之间交换的信息。

未经授权的访问可能来自公司内部，或者如果公司从公司外连接到 extranet 或互联网。

以下场景仅介绍一些未授权客户端如何访问目录数据的示例。

Directory 服务器提供的验证方法、密码策略和访问控制机制可以有效防止未经授权的访问。如需更多信息，请参阅以下部分：

- *第 9.4 节 “选择适当的验证方法”*
- *第 9.6 节 “设计密码策略”*
- *第 9.7 节 “设计访问控制”*

9.1.2. 未授权的 Tampering

如果入侵者可以访问目录服务器和客户端应用程序之间的目录或截获通信，则它们有可能会修改（或修改过）目录数据。如果客户端无法再信任数据，或者目录本身无法信任它从客户端收到的修改和查询，则目录服务无需使用。

例如，如果目录无法检测到篡改，攻击者可能会将客户端的请求更改为服务器（或未转发）并将服务器的响应更改为客户端。TLS 和类似技术可以通过在连接的任一端签名信息来解决这个问题。有关将 TLS 与 Directory 服务器搭配使用的详情请参考 第 9.9 节 “保护服务器连接”。

9.1.3. 拒绝服务

在拒绝服务攻击过程中，攻击者的目标是防止目录为其客户端提供服务。例如，攻击者可能会使用所有系统资源，从而防止这些资源被其他人使用。

目录服务器可以通过在分配给特定绑定 DN 的资源上设置限制来防止拒绝服务攻击。有关根据用户的绑定 DN 设置资源限值的更多信息，请参阅 红帽目录服务器管理指南中的"用户帐户管理"一章。

9.2. 分析安全性需求

分析环境和用户，以确定特定的安全需求。第 3 章 设计目录架构 中的站点调查会阐明有关谁可以读取和编写目录中各个数据的基本决策。这些信息形成了安全设计的基础。

实施了安全性的方式还取决于目录服务用于支持业务的方式。为 Intranet 提供服务的目录不需要与支持向互联网打开的 extranet 或电子商务应用程序的目录相同。

如果该目录仅服务于一个内部网，请考虑信息需要哪些级别的访问权限：

- 如何为用户提供和应用程序，并可访问执行其作业所需的信息。
- 如何保护员工或业务方面的敏感数据。

如果该目录服务于 extranet 或支持互联网上的电子商务应用程序，则需要考虑额外的点：

- 如何为客户提供隐私保证。
- 如何保证信息的完整性。

以下部分提供有关分析安全需要的信息。

9.2.1. 确定访问权限

数据分析标识了访问目录服务所需的信息用户、组、合作伙伴、客户和应用程序。

可以通过以下两种方式之一授予权限：

- 尽可能多授予权限，同时仍然保护敏感数据。

  开放方法需要准确确定哪些数据对业务敏感或至关重要。

- 授予每个类别用户完成其作业所需的最小访问权限。

  限制的方法需要几乎了解用户内部的每个类别的信息需求，并且可能对机构以外各用户的信息需求。

指示用于决定访问权限的方法无关，创建一个简单的表，列出机构中类别以及授予每个权限的访问权限。考虑创建一个列出目录中保存敏感数据的表，并针对各个数据，这是保护数据所采取的步骤。

有关检查用户身份的详情，请参考 第 9.4 节 “选择适当的验证方法”。有关限制访问目录信息的详情，请参考 第 9.7 节 “设计访问控制”

9.2.2. 确保数据保密性和完整性

当使用目录支持通过外部网与业务合作伙伴进行交换时，或支持互联网上的电子商务应用程序，请确保数据交换的隐私性和完整性。

有几种方法可以做到这一点：

- 通过加密数据传输。
- 通过使用证书对数据传输进行签名。

有关 Directory 服务器中提供的加密方法的详情，请参考 第 9.6.2.11 节 “密码存储”

有关签名数据的详情，请参考 第 9.9 节 “保护服务器连接”。

有关加密敏感信息的信息，并将其存储在 Directory Server 数据库中，请参考 第 9.8 节 “加密数据库”

9.2.3. 执行常规审计

作为额外的安全措施，执行常规审计，通过检查日志文件和 SNMP 代理记录的信息来验证总体安全策略的效率。

有关 SNMP 的更多信息，请参阅 红帽目录服务器管理指南。有关日志文件和 SNMP 的更多信息，请参阅 红帽目录服务器管理指南。

9.2.4. Security Needs Analysis 示例

本节提供的示例说明了传统 ISP 公司"example.com"如何分析其安全需求。

example.com 的业务是提供 Web 托管和互联网访问。example.com 活动的一部分是托管客户端公司的目录。它还提供对多个单独订阅者的互联网访问。

因此，example.com 在其目录中有三个主要信息：

- example.com 内部信息
- 属于公司客户的信息
- 与个人订阅者相关的信息

example.com 需要以下访问控制：

- 向其自身目录信息提供托管公司（example_a 和 example_b）的目录管理员的访问权限。
- 为托管公司的目录信息实施访问控制策略。
- 为从其家通过 example.com 进行互联网访问的所有单独客户端实施标准访问控制策略。
- 拒绝对 example.com 的公司目录访问所有外部公司。
- 为世界授予 example.com 订阅者目录的读取访问权限。

9.3. 安全方法概述

目录服务器提供多种设计符合特定需求的总体安全策略的方法。安全策略应该足够强大，以防止未经授权用户修改或检索敏感信息，而且足以便于管理。复杂的安全策略会导致错误，导致人们无法访问或更糟糕的信息，允许人们修改或检索它们不应该被访问的目录信息。

表 9.1. 目录服务器中可用的安全方法

安全方法	描述
身份验证	供一个方验证其他身份的方法。例如，客户端在 LDAP 绑定操作期间为 Directory 服务器提供密码。

安全方法	描述
密码策略	定义密码必须满足的条件，如年龄、长度和语法等。
Encryption	保护信息隐私。当数据被加密时，它会以仅接收者可以理解的方式进行评分。
Access control	定制授予不同目录用户的访问权限，并提供指定所需凭证或绑定属性的方法。
帐户取消激活	禁用用户帐户、帐户组或整个域，以便自动拒绝所有身份验证尝试。
安全连接	通过加密 TLS、启动 TLS 或 SASL 连接来保持信息的完整性。如果在传输过程中加密信息，则接收者可决定在传输过程中不会修改它。设置最低安全强因素，从而需要安全连接。
Auditing	确定目录的安全性是否已被破坏，一个简单的审核方法是查看由 目录维护的日志文件。
SELinux	使用 Red Hat Enterprise Linux 计算机上的安全策略来限制并控制对目录服务器文件和流程的访问。

合并了在安全设计中维护安全性的各种工具，并纳入目录服务的其他功能，如复制和数据分发，以支持安全设计。

9.4. 选择适当的验证方法

有关安全策略的基本决定是用户如何访问该目录。匿名用户访问该目录，或者每个用户都需要使用用户名和密码登录目录（验证）？

目录服务器提供以下身份验证方法：

- 第 9.4.1 节 “匿名和未验证的访问”
- 第 9.4.2 节 “简单绑定和安全绑定”
- 第 9.4.3 节 “基于证书的身份验证”
- 第 9.4.4 节 “代理身份验证”
- 第 9.4.6 节 “无密码验证”

目录对于所有用户使用相同的验证机制，无论用户还是 LDAP 感知型应用程序。

有关防止客户端或一组客户端验证的详情，请参考 第 9.5 节 “设计帐户锁定策略”。

9.4.1. 匿名和未验证的访问

匿名访问提供了目录最简单的访问形式。它使数据可供 目录的任何用户使用，无论它们是否通过身份验证。

但是，匿名访问不允许管理员跟踪谁正在执行哪种搜索类型，而只有某人正在执行搜索。通过匿名访问，连接到目录的任何人都可以访问这些数据。

因此，管理员可能试图阻止特定用户或组访问某些类型的目录数据，但如果允许匿名访问数据，则用户仍然可以通过匿名绑定到目录来访问数据。

匿名访问可能会被限制。通常目录管理员仅允许匿名访问读、搜索和比较特权（不适用于写入、添加、删除或自我写入）。通常，管理员限制对包含通用信息（如姓名、电话号码和电子邮件地址）的属性子集的访问。不允许匿名访问更多的敏感数据，如政府身份号（如美国的社交安全号）、主页电话号码和地址以及 salary 信息。

如果需要通过更严格的规则访问目录数据，也可以完全禁用匿名访问。

当用户试图使用用户名绑定但没有用户密码属性时，未经身份验证的 绑定是。例如：

```
ldapsearch -x -D "cn=jsmith,ou=people,dc=example,dc=com" -b "dc=example,dc=com" "
(cn=joe)"
```

如果用户没有尝试提供密码，目录服务器会授予匿名访问权限。未经身份验证的绑定不需要绑定 DN 为现有条目。

与匿名绑定一样，可以通过限制数据库访问来禁用未经身份验证的绑定来提高安全性。禁用未经身份验证的绑定具有另一个优点：可用于防止客户端的绑定失败。编写较差的应用程序可能会认为它成功通过目录的验证，因为它收到了一个绑定成功的信息，但实际情况是它传递密码失败，只是以未经身份验证绑定的形式进行了简单连接。

9.4.2. 简单绑定和安全绑定

如果不允许匿名访问，用户必须对该目录进行身份验证，然后才能访问目录的内容。借助简单的密码身份验证，客户端通过发送可重复使用的密码向服务器进行身份验证。

例如，客户端使用绑定操作（其提供可区分名称和一组凭证）向 目录进行身份验证。服务器在目录中查找与客户端 DN 对应的条目，并检查客户端给出的密码是否与以该条目存储的值匹配。如果存在，服务器会验证客户端。如果没有，身份验证操作会失败，客户端会收到错误消息。

绑定 DN 通常与一个人的条目对应。但是，有些目录管理员发现，绑定作为组织条目而不是一个人很有用。目录需要绑定的条目是允许 **userPassword** 属性的对象类。这样可确保目录识别绑定 DN 和密码。

大多数 LDAP 客户端从用户隐藏绑定 DN，因为用户可能会发现要记住的 DN 字符的长字符串。当客户端尝试从用户隐藏绑定 DN 时，它使用绑定算法，例如：

1. 用户输入唯一标识符，如用户 ID （例如 **fchen**）。
2. LDAP 客户端应用搜索该标识符的目录，并返回相关的可分名称（如 **uid=fchen,ou=people,dc=example,dc=com**）。
3. LDAP 客户端应用使用检索到的区分名称和用户提供的密码绑定到目录。

简单的密码身份验证提供了一种简单的方法来验证用户，但它需要安全使用额外的安全性。考虑将其用于组织内部网的情况。要通过外部网与业务合作伙伴之间的连接，或用于通过 Internet 与客户进行传输，最好需要安全（加密）连接。

注意

简单密码验证的缺陷是以纯文本形式发送密码。如果未授权用户正在侦听，这可能会破坏目录的安全性，因为该用户能够模仿授权用户。

nsslapd-require-secure-binds 配置属性需要使用 TLS 或 Start TLS 在安全连接中进行简单的密码身份验证。这会有效地加密纯文本密码，使其不能被黑客嗅探。

当使用 TLS 或 Start TLS 操作在 Directory 服务器和客户端应用程序之间建立安全连接时，客户端通过不以纯文本传输密码来执行一个简单的绑定，并带有额外的保护级别。nsslapd-require-secure-binds 配置属性需要通过安全连接进行简单的密码身份验证，即 TLS 或 Start TLS。此设置还支持其他安全连接，如 SASL 身份验证或基于证书的验证。

有关安全连接的详情请参考 第 9.9 节 “保护服务器连接”。

9.4.3. 基于证书的身份验证

另一种形式的目录身份验证涉及使用数字证书绑定到目录。当用户首次访问密码时，该目录会提示用户输入密码。但是，密码不会与 目录中存储的密码不匹配，而是会打开用户的证书数据库。

如果用户提供正确的密码，目录客户端应用程序会从证书数据库获取身份验证信息。然后，客户端应用程序和目录使用此信息通过将用户的证书映射到目录 DN 来识别用户。目录允许或拒绝访问此身份验证过程中确定的目录 DN。

有关证书和 TLS 的更多信息，请参阅 管理指南。

9.4.4. 代理身份验证

代理身份验证是一种特殊的身份验证形式，因为请求访问该目录的用户没有绑定到自己的 DN，而是使用 proxy DN。

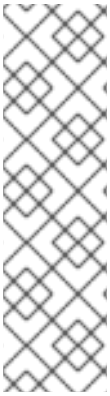
代理 DN 是一个实体，它有权执行用户请求的操作。将代理权限授予个人或应用程序时，他们被授予将任何 DN 指定为代理 DN 的权利，但目录管理器 DN 除外。

代理正确的一个主要优点是，可以启用 LDAP 应用程序，使用单一线程与单一绑定来服务对目录服务器的请求。客户端应用使用代理 DN 绑定到 Directory 服务器，而不是为每个用户绑定和验证。

代理 DN 在由客户端应用程序提交的 LDAP 操作中指定。例如：

```
ldapmodify -D "cn=Directory Manager" -W -x -D "cn=directory manager" -W -p 389 -h
server.example.com -x -Y "cn=joe,dc=example,dc=com" -f mods.ldif
```

此 ldapmodify 命令为 manager 条目(cn=Directory Manager)提供名为 Joe (cn=joe)的权限，以在 mods.ldif 文件中应用修改。管理器不需要提供 Joe 的密码才能进行此更改。



注意

代理机制非常强大，必须完全使用。代理权限在 ACL 的范围内授予，无法限制哪些人被代理右边的输入模拟。也就是说，当用户被授予代理权限时，该用户能够为目标中的任何用户进行代理；无法将代理权限限制为仅限特定用户。

例如，如果实体具有对 dc=example,dc=com 树的代理权限，则该实体可以执行任何操作。因此，请确保在 DIT 的最低可能级别上设置代理 ACI。

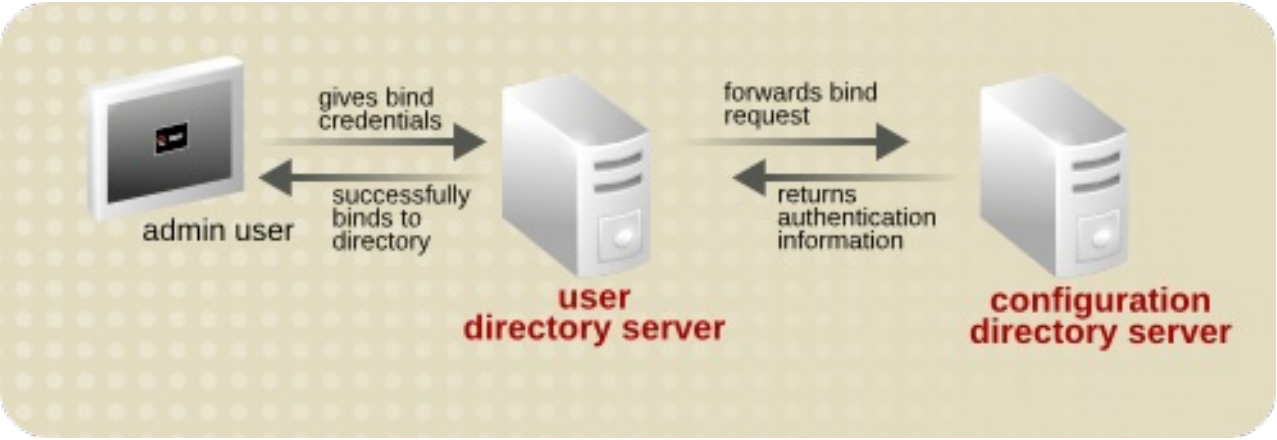
有关此主题的更多信息，请参阅《管理指南》中的"管理访问控制"一章中的"Proxied Authorization ACI 示例"一节。

9.4.5. 直通身份验证

传递身份验证时，任何身份验证请求从一个服务器转发到另一个服务。

例如，每当实例的所有配置信息都存储在另一个目录实例中时，Directory 服务器都会通过身份验证来进行 User Directory 服务器连接配置目录服务器。使用 PTA 插件处理目录服务器到目录服务器传递身份验证。

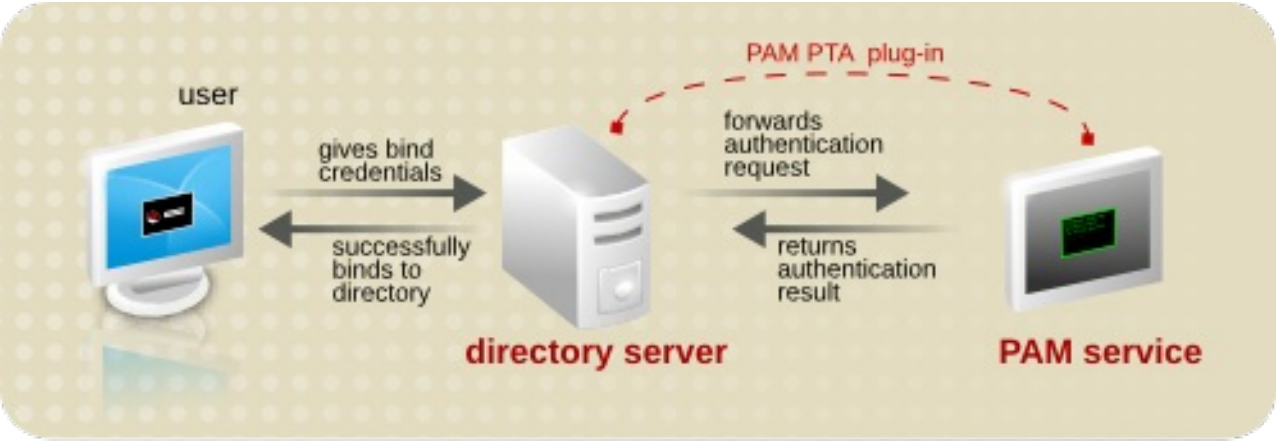
图 9.1. 简单的传递身份验证过程



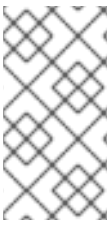
对于 Unix 和 Linux 用户，许多系统已经有身份验证机制。最常用的验证框架之一就是 可插拔验证模块 (PAM)。由于许多网络已经有现有的身份验证服务，因此管理员可能希望继续使用这些服务。可将 PAM 模块配置为告知 Directory 服务器为 LDAP 客户端使用现有身份验证存储。

Red Hat Directory Server 中的 PAM 直通身份验证使用 PAM 传递身份验证插件，这使得目录服务器能够与 PAM 服务通信来验证 LDAP 客户端。

图 9.2. PAM 直通身份验证过程



使用 PAM 传递身份验证时，当用户尝试绑定到 Directory 服务器时，凭证会转发到 PAM 服务。如果凭据与 PAM 服务中的信息匹配，用户可以成功绑定到 Directory 服务器，且所有 Directory 服务器访问控制限制和帐户设置都就位。



注意

目录服务器可以配置为使用 PAM，但无法使用它来 设置 PAM 以使用目录服务器进行身份验证。要使 PAM 使用 Directory 服务器实例进行身份验证，必须正确配置 **pam_ldap** 模块。有关 **pam_ldap** 的常规配置信息，请查看 man page （如 http://linux.die.net/man/5/pam_ldap）。

PAM 服务可使用 系统安全服务守护进程 (SSSD) 等系统工具进行配置。SSSD 可以使用各种不同的身份提供程序，包括 Active Directory、Red Hat Directory Server 或其他目录，如 OpenLDAP 或本地系统设置。要使用 SSSD，只需将 PAM Pass-through Authentication 插件指向 SSSD 使用的 PAM 文件，默认

为 /etc/pam.d/system-auth。

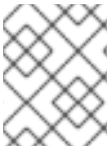
9.4.6. 无密码验证

身份验证尝试评估，首先，用户帐户是否能够进行身份验证。帐户必须处于活跃状态，它不能被锁定，且必须根据适用的密码策略具有有效的密码（也就是说无法过期或需要重置）。

有时，对一个用户是否应该允许进行验证的评估需要被执行，但用户不应该实际被绑定到目录服务器。例如，系统可能使用 PAM 管理系统帐户，而 PAM 则配置为使用 LDAP 目录作为其身份存储。但是，该系统使用的是无密码凭证，如 SSH 密钥或 RSA 令牌，并且这些凭据无法传递给目录服务器。

Red Hat Directory Server 支持 Account Usability Extension Control for ldapsearches。这个控制会返回有关帐户状态和生效的任何密码策略的信息（例如，需要重置、密码过期警告或密码过期警告，或在密码过期后保留的宽限期数量）- 所有在绑定尝试中返回的信息，但不以该用户身份进行身份验证并绑定到目录服务器。这允许客户端根据 Directory 服务器设置和信息来确定用户是否应该被允许，但实际的身份验证过程是在目录服务器之外执行的。

此控制可用于如 PAM 等系统级服务，以允许免密码登录，这样仍使用 Directory 服务器存储身份，甚至控制帐户状态。



注意

默认可由 Directory Manager 使用帐户长度扩展控制。要允许其他用户使用控制，请在支持的控制条目 oid=1.3.6.1.4.1.42.2.27.9.5.8,cn=features,cn=config 上设置适当的 ACI。

9.5. 设计帐户锁定策略

帐户锁定策略可通过防止未经授权或破坏对该目录的访问来保护目录数据和用户密码。在帐户被锁定或取消激活后，该用户无法绑定到该目录，任何验证操作都失败。

帐户停用是通过操作属性 nsAccountLock 实现的。当条目包含值为 true 的 nsAccountLock 属性时，服务器会拒绝该帐户的绑定尝试。

帐户锁定策略可根据特定的自动条件定义：

- 帐户锁定策略可以与密码策略(第 9.6 节 “设计密码策略”)关联。当用户在指定次数后使用正确的凭证登录时，帐户会被锁定，直到管理员手动解锁它。

  这样可防止尝试通过重复尝试猜测用户密码来进入该目录的攻击。

- 在存在一定时间后，可以锁定帐户。这可用于控制临时用户（如 interns、学员或季节性工作者）的访问，根据帐户创建时间限制访问时间。或者，如果帐户在上一次登录时间不活动一段时间内，可以在激活用户帐户时创建帐户策略。

  基于时间的帐户锁定策略通过帐户策略插件来定义，该策略设定目录的全局设置。可以为不同的过期时间和类型创建多个帐户策略子条目，然后通过服务类应用到条目。

另外，可以手动取消激活单个用户帐户或一组帐户（通过角色）。

注意

取消激活角色会取消激活该角色的所有成员，而不是角色条目本身。有关角色的更多信息，请参阅 第 4.3.2 节 “关于角色”。

9.6. 设计密码策略

密码策略是一组规则，用于控制在给定系统中使用密码的方式。Directory Server 的密码策略指定密码必须满足的条件，如年龄、长度以及用户是否可以重复使用密码。

以下小节提供有关设计声密码策略的更多信息：

- 第 9.6.1 节 “密码策略的工作方式”
- 第 9.6.2 节 “密码策略属性”
- 第 9.6.3 节 “在复制环境中设计密码策略”

9.6.1. 密码策略的工作方式

目录服务器支持细粒度密码策略，这意味着可在子树和用户级别上定义密码策略。这允许在目录树的任意点定义密码策略：

- 整个目录。

  这样的策略称为 全局 密码策略。配置并启用后，该策略将应用到 目录中的所有用户，除了启用了本地密码策略的用户条目和那些启用了本地密码策略的用户条目。

  这可以为所有目录用户定义通用、单一密码策略。

- 目录的特定子树。

  这种策略称为 子树级别 或 本地密码策略。配置并启用后，策略将应用到指定子树下的所有用户。

  在托管环境中，为每个托管公司支持不同的密码策略，而不是对所有托管公司强制执行单个策略。

- 目录的特定用户。

  此类策略 称为用户级别 或 本地密码策略。配置并启用后，策略只会应用到指定用户。

  这可以为不同的目录用户定义不同的密码策略。例如，指定一些用户每天更改其密码，一些用户每月更改，而所有其他用户每六个月更改一次。

默认情况下，Directory 服务器包含与全局密码策略相关的条目和属性，这意味着将相同的策略应用于所有用户。要为子树或用户设置密码策略，请在子树或用户级别添加额外的条目，并启用 cn=config 条目的 nsslapd-pwpolicy-local 属性。此属性充当交换机，打开和关闭精细密码策略。

您可以使用命令行或 Web 控制台更改密码策略。使用 dsconf pwpolicy 命令更改全局策略和 dsconf localpwp 命令，以更改本地策略。有关设置密码策略的更多信息，请参阅管理指南。

注意

之前管理的本地密码策略的 ns-newpwpolicy.pl 脚本已弃用。但是，此脚本在 389-ds-base-legacy-tools 软件包中仍然可用。

在将密码策略条目添加到目录中后，它们决定了 Directory 服务器应强制执行的密码策略的类型（global 或 local）。

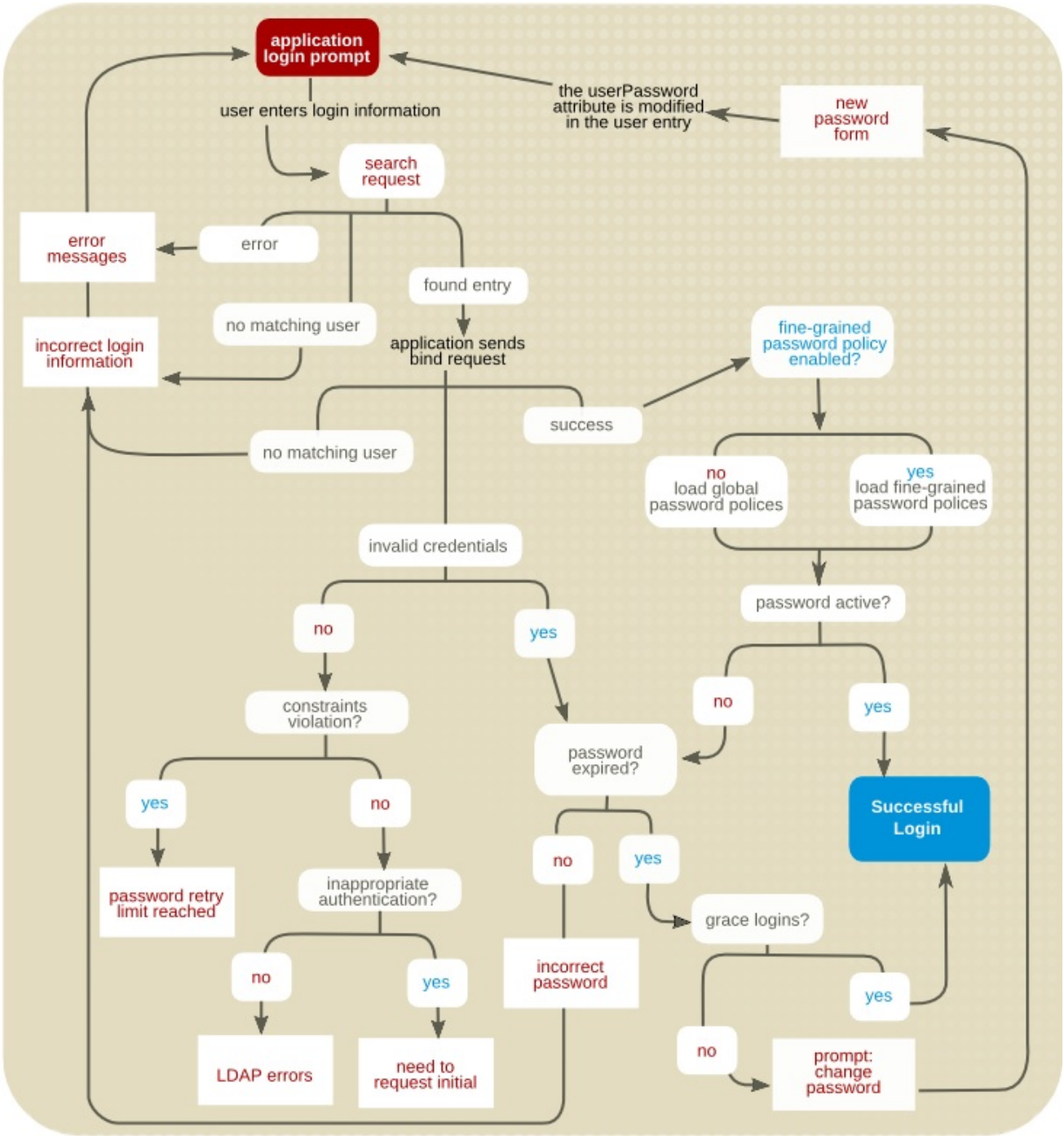
当用户尝试绑定到目录时，Directory 服务器会确定是否为用户条目定义并启用本地策略。

- *要确定是否启用了精细的密码策略，服务器会检查分配给 cn=config 条目的 nsslapd-pwpolicy-local 属性的值(on 或 off)。如果该值为 off，服务器会忽略子树和用户级别定义的策略，并强制实施全局密码策略。*

- *要确定是否为子树或用户定义了本地策略，服务器会在对应的用户条目中检查 pwdPolicysubentry 属性。如果属性存在，服务器会强制为用户配置本地密码策略。如果缺少 属性，服务器会记录错误消息并强制执行全局密码策略。*

然后，服务器会将用户提供的密码与用户的主目录条目中指定的值进行比较，以确保它们匹配。服务器也使用密码策略定义的规则来确保密码在允许用户绑定到目录之前有效。

图 9.3. 密码策略检查过程



除了绑定请求外，如果请求中存在 userPassword 属性（在以下部分中介绍），则密码策略检查也会在添加和修改操作过程中发生。

修改 userPassword 的值检查两个密码策略设置：

- 激活密码最短期限策略。如果最短期限要求尚未满足，服务器会返回 constraintViolation 错误。密码更新操作失败。
- 密码历史记录策略已激活。如果 userPassword 的新值位于密码历史记录中，或者它与当前密码相同，则服务器会返回 constraintViolation 错误。密码更新操作失败。

为密码语法添加和修改 userPassword 检查密码策略的值：

- 激活密码最小长度策略。如果 userPassword 的新值小于所需的最小长度，服务器会返回 constraintViolation 错误。密码更新操作失败。
- 密码语法检查策略已激活。如果 userPassword 的新值与条目的另一个属性相同，服务器会返回 constraintViolation 错误。密码更新操作失败。

9.6.2. 密码策略属性

以下小节描述了为服务器创建密码策略的属性：

- 第 9.6.2.1 节 “最大故障数”
- 第 9.6.2.2 节 “重置后更改密码”
- 第 9.6.2.3 节 “用户定义的密码”
- 第 9.6.2.4 节 “密码过期”
- 第 9.6.2.5 节 “过期警告”
- 第 9.6.2.6 节 “宽限期登录限制”
- 第 9.6.2.7 节 “密码语法检查”
- 第 9.6.2.8 节 “密码长度”
- 第 9.6.2.9 节 “password Minimum Age”
- 第 9.6.2.10 节 “密码历史”
- 第 9.6.2.11 节 “密码存储”
- 第 9.6.2.12 节 “密码最后修改时间”

有关如何设置这些属性的说明，请参阅 红帽目录服务器管理指南。

9.6.2.1. 最大故障数

这是密码策略中的一个设置，可启用基于密码的帐户锁定。如果用户尝试登录某个次数并失败，那么该帐户会锁定，直到管理员解锁或有选择地通过了一定时间。这在 passwordMaxFailure 参数中设置。

当达到最大失败尝试次数时，在评估时可以通过两种不同的方式计算登录尝试。它可以是硬的限制，在数字达到达到时锁定帐户(n)，或仅在超过计数时锁定帐户(n+1)。例如，如果限制为三次失败的尝试，则可以在第三个失败尝试发生时锁定帐户 (n)，或在第四次失败尝试时 (n+1) 锁定账户。n+1 行为是 LDAP 服务器的历史行为，因此它被视为旧行为。较新的 LDAP 客户端预期更严格的硬限制。默认情况下，Directory 服务器使用严格的限制(n)，但可以在 passwordLegacyPolicy 参数中启用旧行为。

9.6.2.2. 重置后更改密码

Directory Server 密码策略可以指定用户是否必须在第一次登录时或管理员重置密码后更改密码。

管理员设置的默认密码通常遵循公司惯例，如用户最初、用户 ID 或公司名称。如果发现这个惯例，这通常是攻击者试图破坏系统的第一个值。因此，建议用户在管理员重置后更改密码。如果为密码策略配置了这个选项，则即使禁用了用户定义的密码，用户需要更改密码。

如果不需要用户或允许更改自己的密码，则管理员分配的密码不应遵循任何明显的惯例，应该很难发现。

默认配置不需要用户在重置后更改密码。

如需更多信息，请参阅 第 9.6.2.3 节 “用户定义的密码”。

9.6.2.3. 用户定义的密码

可以设置密码策略，以允许或不允许用户更改自己的密码。良好的密码是强密码策略的关键。良好的密码不使用微小的词语；在字典、pet 或子代、生日、用户 ID 或任何可方便地发现的用户的其他信息中都可以为密码选择。

好的密码应包括字母、数字和特殊字符的组合。但为了方便起见，用户通常使用容易记住的密码。因此，一些企业选择为用户设置密码，以满足强大密码条件，也不允许用户更改密码。

为用户设置密码有两个缺陷：

- 它需要大量管理员的时间。
- 因为管理员指定的密码通常更难以记住，用户更有可能写出密码，从而增加发现的风险。

默认情况下，允许用户定义的密码。

9.6.2.4. 密码过期

密码策略可允许用户无限期地使用相同的密码，或指定在给定时间后过期的密码。通常，使用较长的密码会被发现。但是，如果密码经常过期，用户可能会发现他们无法记住，并有助于使用密码使用密码。常见策略是使密码每 30 到 90 天过期。

即使禁用密码过期时间，服务器也会记住密码过期规格。如果重新启用了密码过期，则密码只在最后一次禁用前设定的时间有效。

例如，如果将密码策略设置为每隔 90 天过期一次，并且禁用密码过期并重新启用密码过期时间，则默认的密码过期持续时间为 90 天。

默认情况下，用户密码永不过期。

9.6.2.5. 过期警告

如果设置了密码过期周期，最好会在其密码过期前向用户发出警告。

当用户绑定到服务器时，Directory 服务器会显示警告。如果启用了密码过期，默认情况下，会在用户的密码过期之前向用户发送一个警告（使用 LDAP 消息），只要用户的客户端应用程序支持此功能。

要发送的密码过期警告的有效范围为从一到 24,855 天。

注意

密码从未过期直到过期警告被发送为止。

9.6.2.6. 宽限期登录限制

过期密码的宽限期表示用户仍然可以登录到系统，即使其密码已过期。要允许某些用户使用过期密码登录，请在密码过期后指定允许用户的宽限期尝试次数。

默认情况下不允许使用宽限期。

9.6.2.7. 密码语法检查

密码语法检查强制使用密码字符串的规则，以便任何密码都必须满足或超过特定条件。所有密码语法检查都可在全局、每个子树或每个用户应用。密码语法检查在 passwordCheckSyntax 属性中设置。

默认密码语法要求最小密码长度为八个字符，且密码中没有使用任何普通词语。微小的词语是存储在 uid、cn、sn、gedName、gou、nou 或 mail 属性中的任何值。

另外，有可能进行其他形式的密码语法强制，为密码语法提供不同的可选类别：

- 密码中所需的最小字符数(passwordMinLength)
- 最小数字数，即零到 9 之间的数字(passwordMinDigits)
- 大写和小写的 ASCII 字母字符数(passwordMinAlphas)
- 最少大写 ASCII 字母字符数(passwordMinUppers)
- 最低小写 ASCII 字母数(passwordMinLowers)
- 最小特殊 ASCII 字符数，如 !@#$ (passwordMinSpecials)
- 最小 8 位字符数(passwordMin8bit)
- 可以立即重复同一字符的次数上限，如 aaabbb (passwordMaxRepeats)
- 每个密码所需的最小字符类别数；类别可以是大写或小写字母、特殊字符、数字或 8 位字符 (passwordMinCategories)

- 目录服务器针对 CrackLib 字典检查密码(passwordDictCheck)
- 目录服务器检查密码是否包含 palindrome (passwordPalindrome)
- 目录服务器可防止设置同一类别(passwordMaxClassChars)中连续字符的密码。
- 目录服务器可防止设置包含某些字符串的密码(passwordBadWords)
- 目录服务器可防止设置包含管理员定义属性中设置的字符串的密码 (passwordUserAttributes)

需要的更多语法类别，其密码越高。

默认情况下禁用密码语法检查。

9.6.2.8. 密码长度

密码策略可能需要用户密码的最短长度。通常，更短的密码更易于破解。密码的良好长度为 8 个字符。这很难破解但很短，用户可以在不写出密码的情况下记住密码。此属性的有效范围值从 2 到 512 个字符。

默认情况下不设置密码长度。

9.6.2.9. password Minimum Age

密码策略可以阻止用户更改指定时间的密码。与 passwordHistory 属性一起使用时，不建议用户使用旧密码。

例如，如果密码最短年龄(passwordMinAge)属性是 2 天，用户可以在单个会话中重复更改其密码。这可以防止它们通过密码历史记录加以利用，以便他们可以重复利用旧密码。

此属性的有效范围值从零到 24,855 天。值为零(0)表示用户可以立即更改密码。

9.6.2.10. 密码历史

目录服务器可以在密码历史记录中存储两个到 24 密码 ；如果密码位于历史记录中，用户也无法将其密码重置为旧密码。这可防止用户重复使用容易记住的一些密码。或者，可以禁用密码历史记录，从而允许用户重复使用密码。

即使密码历史记录已关闭，密码历史记录也仍处于历史记录中，用户在禁用密码历史记录前无法重复使用历史记录中的密码。

默认情况下，服务器不会维护密码历史记录。

9.6.2.11. 密码存储

密码存储方案指定了在目录中存储目录服务器密码的加密类型。Directory 服务器支持几种不同的密码存储方案：

- salt 的安全哈希算法 （SSHA、SSHA-256、SSHA-384 和 SSHA-512）。这是最安全的密码存储方案，它是默认的。 推荐的 SSHA 方案是 SSHA-256 或 stronger。
- CLEAR，这意味着没有加密。这是唯一可与 SASL Digest-MD5 一起使用的选项，因此使用 SASL 需要 CLEAR 密码存储方案。

  虽然目录中存储的密码可以通过使用访问控制信息 (ACI) 指令进行保护，但它仍然不是将纯文本密码存储在目录中的最佳选择。
- 安全散列算法 （SHA、SHA-256、SHA-384 和 SHA-512）。这比 SSHA 更安全。
- UNIX CRYPT 算法.这个算法提供与 UNIX 密码的兼容性。
- MD5.这个存储方案比 SSHA 不太安全，但对于需要 MD5 的传统应用程序会包括这个存储方案。
- salt 的 MD5。这个存储方案比普通 MD5 哈希更安全，但仍然比 SSHA 更安全。此存储方案不包含用于新密码，而是有助于将用户帐户从支持 salt 的 MD5 的目录中迁移。

9.6.2.12. 密码最后修改时间

passwordTrackUpdateTime 属性告知服务器最后一次为条目更新密码的时间戳。密码更改时间本身作为操作属性存储在用户条目 pwdUpdateTime 上（与 modifyTimestamp 或 lastModified 操作属性分开）。

默认情况下，不会记录 密码更改时间。

9.6.3. 在复制环境中设计密码策略

在复制环境中强制实施密码和帐户锁定策略，如下所示：

- 在数据供应商强制密码策略。
- 在复制设置中的所有服务器上强制使用帐户锁定。

目录中的密码策略信息（如密码年龄）、帐户锁定计数器和过期警告计数器都是复制的。但是，配置信息不存储在本地，且不会被复制。此信息包括密码语法以及密码修改的历史记录。

在复制环境中配置密码策略时，请考虑以下点：

- 所有副本都发出警告密码过期。此信息保留在每台服务器上，因此，如果用户依次绑定到多个副本，则用户会多次收到相同的警告。此外，如果用户更改密码，则可能需要过些时间，将此信息过滤到副本。如果用户更改密码，然后立即重新绑定，绑定可能会失败，直到副本注册更改为止。
- 所有服务器上都应该发生相同的绑定行为，包括供应商和副本。始终在每台服务器上创建相同的密码策略配置信息。
- 在多层次环境中，帐户锁定计数器可能无法按预期工作。

9.7. 设计访问控制

在决定用来建立目录客户端身份的身份验证方案后，决定如何使用这些方案来保护目录中包含的信息。访问控制可指定某些客户端可以访问特定信息，而其他客户端则不能访问。

使用一个或多个 访问控制列表 (ACL) 定义访问控制。目录的 ACL 包括一个或多个 访问控制信息 (ACI) 语句，允许或拒绝权限（如读取、写入、搜索和比较）与指定的条目及其属性。

使用 ACL，可以在目录树的任何级别上设置权限：

- 整个目录。
- 目录的特定子树。
- 目录中特定条目。
- 一组特定条目属性。
- 任何与给定 LDAP 搜索过滤器匹配的条目。

此外，可以为特定用户、属于特定组的所有用户或目录所有用户设置权限。最后，可以为网络位置（如 IP 地址（IPv4 或 IPv6）或 DNS 名称定义访问权限。

9.7.1. 关于 ACI 格式

在设计安全策略时，了解在目录中如何表示 ACI 会很有帮助。理解可以在目录中设置什么权限。本节概述了 ACI 机制。有关 ACI 格式的完整描述，请参阅 Red Hat Directory Server Administration Guide。

目录 ACI 使用以下通用表单： 目标权限 bind_rule

ACI 变量定义如下：

- target.指定 ACI 目标、目标属性或两个条目（通常是子树）。目标标识 ACI 应用到的目录元素。ACI 只能将一个条目为目标，但可以针对多个属性。此外，目标也可以包含 LDAP 搜索过滤器。可以为包含通用属性值的广泛分散条目设置权限。

- 权限。标识此 ACI 所设置的实际权限。权限变量指出，ACI 被允许或拒绝特定类型的目录访问，如读取或搜索，到指定的目标。

- 绑定规则。标识权限应用到的绑定 DN 或网络位置。绑定规则也可以指定 LDAP 过滤器，如果该过滤器被评估为绑定客户端应用程序，则 ACI 应用到客户端应用程序。

因此，ACI 可以表达如下："对于目录对象目标，如果 bind_rule 为 true，则允许或拒绝权限"。

权限 和 bind_rule 被设置为一个对，可以有多个 权限- 每个目标的bind_rule 对。可以有效地为任何给定目标设置多个访问控制。例如：

> target (permission bind_rule)(permission bind_rule)...

可以设置权限，以允许任何人绑定 Babs Jensen 写入 Babs Jensen 的电话号码。此权限中的绑定规则是说明 "如果您绑定为 Babs Jensen"的部分。目标是 Babs Jensen 的电话号码，权限是 写访问权限。

9.7.1.1. 目标

决定在目录中创建的每个 ACI 的目标条目。定位目录分支点条目包括该分支点及其在权限范围内的所有子条目。如果没有为 ACI 明确定义目标条目，则 ACI 的目标是包含 ACI 语句的目录条目。将 targetattr 参数设置为以一个或多个属性为目标。如果没有设置 targetattr 参数，则不会作为任何属性。详情请查看 Red Hat Directory Server Administration Guide 中的对应部分。

对于每个 ACI，每个 ACI 只能将一个条目或与单个 LDAP 搜索过滤器匹配的条目作为目标。

除了目标条目外，该条目上的目标属性也可以应用权限，这只对属性值的子集应用。通过显式命名这些属性或明确指定不以 ACI 的目标属性来命名这些属性的目标属性。目标中排除属性可为所有设置权限，但对象类结构允许的一些属性。

详情请查看 Red Hat Directory Server Administration Guide 中的对应部分。

9.7.1.2. 权限

权限可以允许或拒绝访问。通常，避免拒绝权限（因为 第 9.7.2.2 节 “允许或拒绝访问”中介绍的原因）。权限可以是在目录服务中执行的任何操作：

权限	描述
读	指明目录数据是否可以读取。
写	指明目录数据是否可以更改或创建。此权限还允许删除目录数据，但不能删除条目本身。要删除整个条目，用户必须具有删除权限。
搜索	指明能否搜索目录数据。这与读取权限不同，如果作为搜索操作的一部分返回，读取允许查看目录数据。 例如，如果搜索常用名称以及个人房间编号的读取权限，则可以返回房间号码作为通用名称搜索的一部分，但空间编号本身不能用作搜索的主题。使用这个组合以防止人员搜索目录，以查看谁位于特定房间。
比较	指明数据是否可以用于比较操作。比较权限表示搜索功能，但搜索实际目录信息不会返回。相反，会返回一个简单的布尔值，指明比较的值是否匹配。这用于在目录身份验证过程中匹配 userPassword 属性值。
自我写入	仅用于组管理。这个权限可让用户在组中添加或删除自己。
添加	指明是否可以创建子条目。这个权限可让用户在目标条目的下面创建子条目。
删除	指明是否可以删除条目。这个权限可让用户删除目标条目。
Proxy	表示用户可以使用任何其他 DN（ Directory Manager 除外）访问具有此 DN 权利的目录。

9.7.1.3. 绑定规则

绑定规则通常表示绑定 DN 受该权限的影响。它还可以指定绑定属性，如一天或 IP 地址的时间。

绑定规则轻松表达 ACI 仅适用于用户自己的条目。这允许用户在更新另一个用户条目的情况下更新自己的条目。

绑定规则表示 ACI 适用于特定情况：

- 只有绑定操作是从特定 IP 地址（IPv4 或 IPv6）或 DNS 主机名到达时。这通常用于强制从给定计算机或网络域进行所有目录更新。
- 如果个人匿名绑定。为匿名绑定设置权限也意味着，权限也适用于绑定到该目录的任何人。
- 对于成功绑定到该目录的任何人。这可在阻止匿名访问时进行常规访问。
- 只有客户端绑定为条目的直接父项时。
- 只有该人绑定了满足特定 LDAP 搜索条件的条目。

目录服务器提供多个关键字来更轻松地表达这类访问：

- 父.如果 bind DN 是即时父条目，则绑定规则为 true。这意味着，可以授予特定权限，允许目录分支点管理其即时子条目。
- 自我.如果 bind DN 与请求访问的条目相同，则绑定规则为 true。可以授予特定权限以允许个人更新他们自己的条目。
- 所有.对于成功绑定到该目录的任何人，绑定规则为 true。
- Anyone.对于每个人，绑定规则都为 true。这个关键字用于允许或拒绝匿名访问。

9.7.2. 设置权限

默认情况下，除 Directory Manager 外，所有用户都被拒绝访问任何类型的权限。因此，必须为目录设置一些 ACI，以便用户可以访问该目录。

有关如何在目录中设置 ACI 的详情，请参考 Red Hat Directory Server Administration Guide。

9.7.2.1. Precedence Rule

当用户尝试任何对目录条目的访问时，Directory 服务器检查目录中设置的访问控制。要确定访问权限，目录服务器应用 优先级规则。此规则指出，当存在两个冲突的权限时，拒绝访问的权限优先于授予访问权限的权限。

例如，如果在目录的根目录下拒绝写入权限，并且该权限适用于访问该目录的任何人，无论可能允许写入任何权限，任何用户都可对该目录进行写入。要允许特定用户对该目录的写入权限，必须设置原始 deny-for-write 范围，使其不会包含该用户。然后，有疑问的用户必须有额外的 allow-for-write 权限。

9.7.2.2. 允许或拒绝访问

对目录树的访问可以明确允许或拒绝，但要谨慎拒绝对该目录的访问。由于具有优先级规则，如果目录明确找到访问规则，无论可能授予了任何冲突权限，则用于禁止访问的目录。

限制允许访问规则的范围只包括用户或客户端应用程序的最小可能子集。例如，可以设置权限，允许用户在其目录条目上写入任何属性，但拒绝除 Directory 管理员组成员以外的所有用户写入 uid 属性的权限。或者，以以下方式编写允许写入访问的两个访问规则：

- 创建一个规则，允许向每个属性写入特权，但 uid 属性除外。此规则应适用于每个人。
- 创建一个规则，允许对 uid 属性进行写入特权。此规则应仅应用于 Directory 管理员组的成员。

仅允许特权，避免设置显式拒绝特权。

9.7.2.3. 当拒绝访问

很少需要设置显式拒绝权限，但在有些情况下，它很有用：

- 有一个大型目录树，其中包括复杂的 ACL。

  为安全起见，可能需要突然拒绝对特定用户、组或物理位置的访问。不必花费时间仔细检查现有的 ACL 以了解如何适当限制允许权限，而是临时设置显式拒绝特权，直到有时间进行分析。如果 ACL 变得很复杂，对于长远来讲，使用拒绝 ACI 只会添加管理的开销。在可能的情况下，尽快取消相关的 ACL 以避免显式拒绝特权，然后简化整体访问控制方案。

- 访问控制应该基于周中的一天或一天中的一个小时。

  例如，所有写入活动都可以在下午 11:00 时拒绝。（2300 到周一至周一的上午 1:00.(0100). 从管理的角度来看，管理一个基于时间的 ACI 更容易，其明确限制此类的基于时间的访问，而不是搜索所有 allow-for-write ACIs 并在这一时间段内限制它们的范围。

- 在委派目录管理机构到多个人员时，应限制特权。

  要允许一个人或一组人管理目录树的某些部分，而无需修改树的某些方面，请使用明确拒绝特权。

  例如，要确保 Mail Administrators 不允许对通用 name 属性的写访问，请设置可明确拒绝对常见 name 属性的写入访问权限的 ACI。

9.7.2.4. Place Access Control Rules 的位置

访问控制规则可以放在目录中的任何条目上。通常，管理员将访问控制规则放在带有对象类 domainComponent,country,organization,organizationalUnit,inetOrgPerson, 或 group 的条目上。

将规则组织为组尽可能多，以简化 ACL 管理。规则通常适用于其目标条目和所有条目的子项。因此，最好将访问控制规则放在目录分支点或目录分支点上，而不是分散在个别叶（如个人）条目中。

9.7.2.5. 使用过滤的访问控制规则

Directory Server ACI 模型的其中一个更强大的功能是能够使用 LDAP 搜索过滤器来设置访问控制。使用 LDAP 搜索过滤器，将访问权限设置为符合定义的一组条件的任何目录条目。

例如，允许对包含 organizationalUnit 属性的 entry 的读访问权限允许设置为 marketing。

过滤的访问控制规则允许预定义的访问级别。假设目录包含家地址和电话号码信息。有些人希望发布这些信息，另一些则希望取消列出该信息。解决的方法有几种：

- 在每个用户的目录条目上创建一个名为 publishHomeContactInfo 的属性。

- 设置一个访问控制规则，仅针对其 publishHomeContactInfo 属性设为 true 的条目授予对 homePhone 和 homePostalAddress 属性的读取访问权限。使用 LDAP 搜索过滤器来表达此规则的目标。

- 允许目录用户将自己的 publishHomeContactInfo 属性的值更改为 true 或 false。这样，目录用户可以决定是否公开此信息。

有关使用 LDAP 搜索过滤器以及使用 ACI 的 LDAP 搜索过滤器的更多信息，请参阅 红帽目录服务器管理指南。

9.7.3. 查看 ACI：获取 Effective Rights

可能需要查看条目上的访问控制集，以便授予细粒度访问控制或高效的条目管理。获取有效权限 是一个扩展 ldapsearch，它返回对条目中的每个属性设置的访问控制权限，并允许 LDAP 客户端确定服务器访问控制配置允许用户执行哪些操作。

访问控制信息被分为两组的访问权限：对于一个属性，权限是条目和权利。"条目的右边表示其权限（如修改或删除），仅限于该特定条目。"在属性的右侧表示该属性中每一个实例的访问权限。

在以下情况下可能需要这种详细的访问控制：

- 管理员可以将 get effective rights 命令用于分钟的访问控制，例如允许特定组或用户访问条目并限制其他组。例如，QA Managers 组的成员可以搜索和读取 标题 和 salary 等属性，但只有 HR Group 成员具有修改或删除它们的权限。

- 用户可以使用 get effective rights 选项来确定他们可以在自己的个人条目上查看或修改哪些属性。例如，用户应该有权访问 homePostalAddress 和 cn 等属性，但可能只具有对 title 和 salary 的读取访问权限。

使用 -E 交换机执行的 ldapsearch 返回特定条目的访问控制作为正常搜索结果的一部分。以下搜索显示了用户 Ted Morris 必须对其个人条目的权限：

```
ldapsearch -x -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com"
-W -b "uid=tmorris,ou=people,dc=example,dc=com" -E
!1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=tmorris,ou=people,dc=example,dc=com "(objectClass=*)"

version: 1
dn: uid=tmorris,ou=People,dc=example,dc=com
```

```
givenName: Ted
sn: Morris
ou: Accounting
ou: People
l: Santa Clara
manager: uid=dmiller,ou=People,dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1lOHtMD6yqPyhxBA==
entryLevelRights: vadn
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rscow, manager:rsc,
roomNumber:rscwo, mail:rscwo, facsimileTelephoneNumber:rscwo, objectClass:rsc, uid:rsc,
cn:rsc, userPassword:wo
```

在本例中，Ted Morris 具有在其自己的条目上添加、查看、删除或重命名 DN 的权利，如 entryLevelRights 所示。他可以读取、搜索、比较、自我修改或自我删除位置(l)属性，但仅对其密码进行自我写入和自我删除权限，如 attributeLevelRights 结果所示。

默认情况下，条目中没有值或条目中没有对应的属性不会返回有效的权限信息。例如，如果删除了 userPassword 值，则以后对上述条目的有效权限搜索不会返回 userPassword 的任何有效权限，即使可以允许自助和自我删除权利。同样，如果添加了 street 属性读取、比较和搜索权利，则 street: rsc 将出现在 attributeLevelRights 结果中。

可以返回通常未包含在搜索结果中的属性的权限，如不存在的属性或操作属性。使用星号(*)返回条目所有可能属性的权限，包括不存在的属性。

```
ldapsearch -x -E !1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=scarter,ou=people,dc=example,dc=com ''
(objectclass=*)'' "*"
```

使用加号(+)返回条目的操作属性，它们通常不会在 ldapsearch 星号 packagemanifests 中返回。例如：

```
ldapsearch -x -E !1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=scarter,ou=people,dc=example,dc=com ''
(objectclass=*)'' "+"
```

星号(*)和加号(+)可以一起使用，以返回条目的每个属性。

9.7.4. 使用 ACI：某些 Hint 和 Tricks

在实施安全策略时请注意这一提示。它们有助于降低管理目录安全模型的管理负担，并改进目录的性能特征。

- 最小化目录中的 ACI 数量。

  虽然目录服务器可以评估超过 50,000 ACI，但很难管理大量 ACI 语句。大量 ACI 可让人工管理员立即决定特定客户端可用的目录对象。

  目录服务器使用宏最小化目录中的 ACI 数量。宏是用于在 ACI 中代表 DN 或 DN 的部分占位符。使用 宏代表 ACI 或绑定规则部分或两者的目标部分中的 DN。有关宏 ACI 的更多信息，请参阅红帽目录服务器管理指南中的" 管理访问控制"章节。

- 平衡允许和拒绝权限。

  虽然默认规则是拒绝对任何没有被特别授予访问权限的用户的访问，但最好使用一个命令 ACI 来减少 ACI 的数量，以便访问树的根，以及少量拒绝 ACI 条目。这种情境可避免使用多个 allow ACI 来接近leaf 条目。

- 识别任何给定 ACI 上属性的最小集合。

  在允许或拒绝对象中属性子集的访问时，确定最小列表是允许的属性还是拒绝的属性集合。然后，指定 ACI，以便它只需要管理最小列表。

  例如，person 对象类包含大量属性。要允许用户仅更新其中一个或两个属性，请编写 ACI，以便仅允许这些几个属性进行写入访问。但是，要允许用户更新除一或两个属性以外的所有属性，请创建 ACI，以便它允许对所有内容进行写入访问，但有几个命名的属性。

- 谨慎使用 LDAP 搜索过滤器。

  搜索过滤器不会直接命名管理访问权限的对象。因此，它们的使用可能会导致意外的结果。当目录变得更为复杂时，这尤其如此。在 ACI 中使用搜索过滤器前，使用同一过滤器运行一个 ldapsearch 操作，以明确更改的结果的含义。

- 不要在目录树的不同部分中重复 ACI。

  针对重叠 ACI 的保护。例如，如果目录根点上有一个 ACI，则允许组对 commonName 和 givenName 属性的写入访问权限，而另一个 ACI 则仅允许对 commonName 属性执行同一组写入权限，然后考虑对 ACI 进行修改，以便只有一个控制授予组的写入访问权限。

  随着目录的日益复杂，意外的 ACI 的风险很快会增加。通过避免 ACI 重叠，安全性管理变得更加容易，同时可能会减少包含在目录中的 ACI 总数。

- 名称 ACI.

  在命名 ACI 是可选的时，为每个 ACI 提供简短的、有意义的名称有助于管理安全模型。

- 在目录中尽可能地对 ACI 进行分组。

  尝试将 ACI 放置限制为目录根点和主要目录分支点。对 ACI 进行分组有助于管理 ACI 的总列表，并帮助保持目录中的 ACI 总数最少。

- 避免使用双引号，例如，如果绑定 DN 不等于 cn=Joe 时拒绝写入。

  尽管此语法对于服务器是完全可以接受的，但它对于人类管理员来说令人困惑。

9.7.5. 将 ACI 应用到根 DN（目录管理器）

通常，访问控制规则不适用于 Directory Manager 用户。Directory Manager 在 dse.ldif 文件中定义，而不是在常规用户数据库中定义，因此 ACI 目标不包括该用户。

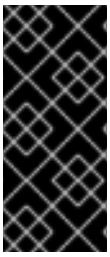
从维护角度来说也有意义。Directory Manager 需要高级别的访问权限来执行维护任务并响应事件。

仍由于 Directory Manager 用户的强大功能，可能会建议某些级别的访问控制来防止未经授权的访问或攻击以 root 用户身份执行。

RootDN Access Control 插件可设置特定于 Directory Manager 用户的特定访问控制规则：

- 基于时间的访问控制用于时间范围，如 8a.m 到 5p.m。（转至 1700）。
- day-of-week 访问控制，因此仅在明确定义的天数时允许访问
- IP 地址规则，其中只有指定的 IP 地址、域或子网被显式允许或拒绝
- 主机访问规则，其中只有指定的主机名、域名或子域被显式允许或拒绝

与其他访问控制规则一样，拒绝规则监管规则。



重要

确保 Directory Manager 始终具有允许的访问级别。Directory Manager 可能需要在非小时内执行维护操作（用户负载较差时），或响应故障。在这种情况下，设置字符串时或基于日期的访问控制规则可能会阻止 Directory Manager 能够充分管理该目录。

默认情况下禁用根 DN 访问控制规则。必须启用 RootDN Access Control 插件，然后可以设置适当的访问控制规则。

注意

在插件条目中，只能为 Directory Manager 设置一个访问控制规则，它适用于所有访问整个目录。

9.8. 加密数据库

信息以纯文本形式存储在数据库中。因此，一些非常敏感的信息（如政府身份号或密码）可能不足受到访问控制措施的保护。有可能获得服务器持久存储文件的访问权限，可以直接通过文件系统或访问丢弃的磁盘驱动器或存档媒体。

通过数据库加密，可以加密个别属性，因为它们存储在数据库中。配置后，特定属性（甚至索引数据）的每个实例都将被加密，且只能使用安全频道（如 TLS）进行访问。

有关使用数据库加密的详情，请参考红帽目录服务器管理指南中的"配置目录数据库"一章。

9.9. 保护服务器连接

在为识别的用户设计验证方案和用于保护目录中信息的访问控制方案后，下一步是为在服务器和客户端应用程序之间传递信息的完整性而设计的方法。

对于客户端连接和服务器连接的服务器，Directory 服务器支持各种安全连接类型：

- 传输层安全性(TLS)

  为了通过网络提供安全通信，目录服务器可以在传输层安全(TLS)上使用 LDAP。

  TLS 可以与来自 RSA 的加密算法一起使用。为特定连接选择的加密方法是客户端应用程序和目录服务器之间的协商结果。

- 启动 TLS。

  目录服务器也支持启动 TLS，这是通过常规的未加密 LDAP 端口发起传输层安全(TLS)连接的方法。

- 简单身份验证和安全层(SASL)

  SASL 是安全框架，这意味着它设置一个系统，允许不同的机制向服务器验证用户，这取决于客户端和服务器应用程序中启用了哪些机制。它还可在客户端和服务器之间建立一个加密的会话。在目录服务器中，SASL 与 GSS-API 一起使用以启用 Kerberos 登录，并可用于几乎所有服务器到服务器连接，包括复制、链接和通过传递身份验证。（SASL 无法与 Windows 同步一起使用。）

对于处理敏感信息（如复制）以及一些操作（如 Windows 密码同步）需要的操作，推荐使用安全连接。目录服务器可以同时支持 TLS 连接、SASL 和非安全连接。

SASL 身份验证和 TLS 连接可以同时配置。例如，可以将 Directory 服务器实例配置为要求 TLS 连接到服务器，并支持复制连接的 SASL 身份验证。这意味着不需要选择在网络环境中使用 TLS 还是 SASL；您可以使用两者。

也可以为与服务器连接设置最低级别的安全性。安全强度因 措施的关键强度，它如何实现安全连接。可以设置 ACI，要求仅在连接特定强度或更高级别时才进行某些操作（如密码更改）。也可以设置最小 SSF，它实际上可以禁用标准连接，并且每个连接都需要 TLS、启动 TLS 或 SASL。Directory 服务器同时支持 TLS 和 SASL，并且服务器计算所有可用连接类型的 SSF 并选择强度。

有关使用 TLS、启动 TLS 和 SASL 的更多信息，请查看管理指南。

9.10. 使用 SELINUX 策略

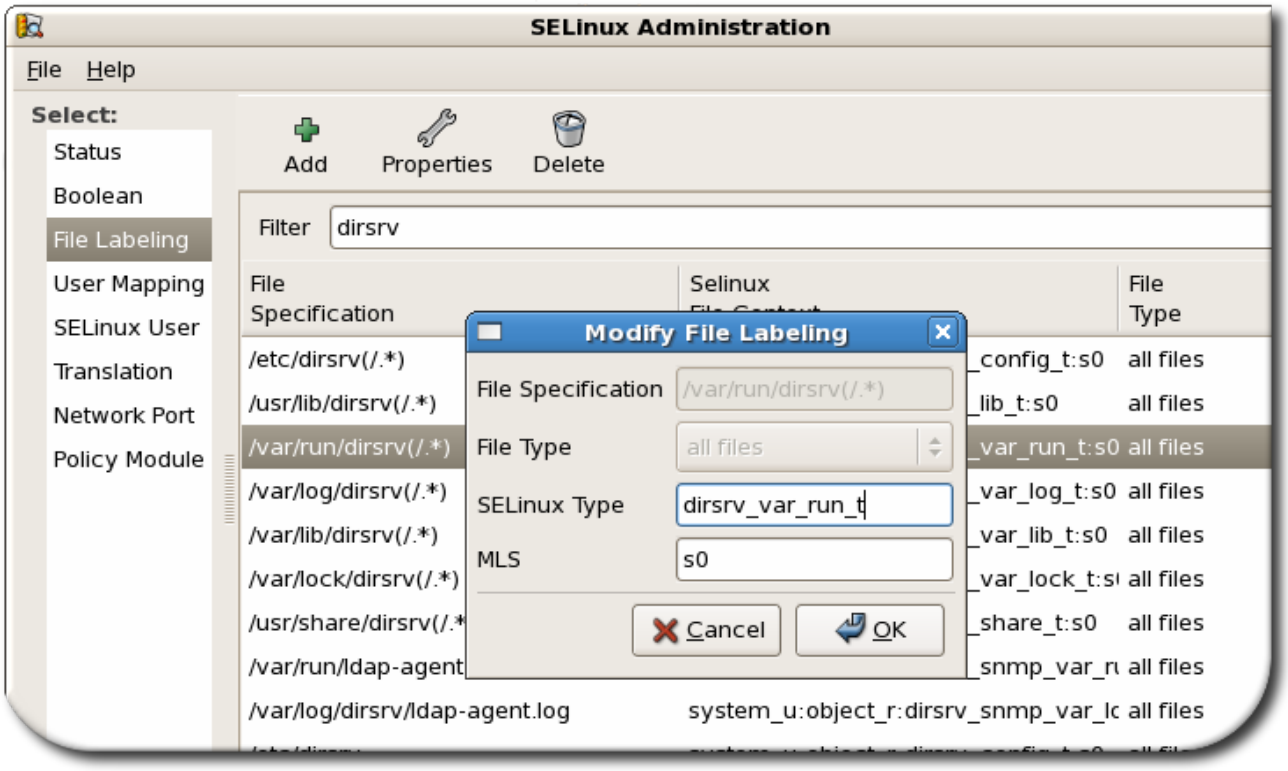
SELinux 是强制访问控制规则的集合，用于限制未经授权的访问和篡改。SELinux 对服务器上的文件、目录、端口、进程、用户和其他对象进行分类。每个对象放置在适当的安全上下文中，以定义对象如何通过角色、用户和安全级别在服务器的行为中。这些角色在域中分组，SELinux 规则定义如何允许一个域中的对象与另一个域中的对象交互。

目录服务器有以下域：

- Directory 服务器的 dirsrv_t
- SNMP 的 dirsrv_snmp_t

目录服务器还使用一个额外的默认域用于 LDAP 端口： ldap_port_t

图 9.4. 编辑目录服务器文件标签



这些域为 Directory 服务器的所有进程、文件、目录、端口、套接字和用户提供安全上下文。

- *每个实例的文件和目录都带有特定的 SELinux 上下文。（Directory Server 使用的大多数主要目录具有所有本地实例的子目录，无论多少次，一个策略都轻松应用到新实例。）*
- *每个实例的端口使用特定的 SELinux 上下文标记。*
- *所有目录服务器进程都在适当的域中进行限制。*
- *每个域具有特定的规则，它们定义域授权了哪些操作。*
- *SELinux 策略中没有指定的访问权限将拒绝实例。*

SELinux 有三个不同的执行级别：禁用（没有 SELinux）、permissive（规则会被处理但不强制这些规则）和 enforcing（所有规则都强制）。Red Hat Directory Server 定义了 SELinux 策略，允许它在严格的 SELinux enforcing 模式下以正常方式运行。目录服务器可以在不同的模式下运行，一个用于正常操作，一个用于导入等数据库操作（ldif2db 模式）。Directory 服务器的 SELinux 策略仅适用于正常模式。

默认情况下，Directory 服务器会受 SELinux 策略限制。

9.11. 其他安全资源

有关设计安全目录的更多信息，请查看以下操作：

- 了解和部署 LDAP 目录服务.T. Howes, M. Smith, G. am, Macmillan Technical Publishing, 1999.
- SecurityFocus.com http://www.securityfocus.com
- 计算机出现响应团队(CERT)协调中心 http://www.cert.org

第 10 章 目录设计示例

目录服务的设计取决于企业的大小和性质。本章提供了几个示例，它演示了如何在各种不同的设置中应用目录。这些示例是开发实时目录服务部署计划的起点。

10.1. 设计示例：本地企业

企业(Amobileile parts manufacturer)是由 500 名员工组成的小公司。示例 Corp. 决定部署红帽目录服务器来支持其所使用的目录应用程序。

10.1.1. 本地企业数据设计

示例。首先确定其将存储在 目录中的数据类型。为此，example Corp. 创建一个部署团队，它将执行站点调查来确定如何使用该目录。部署团队决定以下内容：

- Corp. 的 目录将由消息传递服务器、Web 服务器、日历服务器、人工资源应用程序和白页应用程序使用。
- 消息传递服务器对 uid、mailServerName 和 mailAddress 等属性执行精确搜索。为提高数据库性能，example Corp. 将维护这些属性的索引，以支持通过消息传递服务器搜索。

  有关使用索引的详情请参考 第 6.4 节 “使用索引来提升数据库性能”。
- 空白页面应用程序会频繁搜索用户名和电话号码。因此，该目录需要能够频繁地子字符串、通配符和 fuzzy 搜索，后者返回大量结果。示例 Corp. 决定维护 cn,sn, 和 givenName 属性的存在、相等、大约和子字符串索引，以及 telephoneNumber 属性的 presence, equality, 和 substring 索引。
- 示例 Corp. 的目录维护用户和组信息，以支持整个组织内部署的基于 LDAP 服务器的 Intranet。大多数 Example Corp. 的用户和组信息将由一组目录管理员集中管理。但是，Example Corp. 还希望电子邮件信息由独立的邮件管理员组管理。
- 企业计划将来支持公钥基础架构(PKI)应用程序，如 S/MIME 电子邮件，因此需要准备好将用户的公钥证书存储在目录中。

10.1.2. 本地企业架构设计

Corp. 的部署团队决定使用 inetOrgPerson 对象类来代表目录中的条目。这个对象类是 appealing，因为它允许 userCertificate 和 uid (userID)属性，这两个属性都由 Example Corp. 目录支持。

Corp. 也希望自定义默认目录模式。示例 Corp. 创建 examplePerson 对象类来代表 Example Corp 的员工。它从 inetOrgPerson 对象类派生此对象类。

examplePerson 对象类允许一个属性 exampleID 属性。此属性包含分配给每个 Example Corp. staff 的特殊员工号码。

未来， Example Corp. 可以根据需要向 examplePerson 对象类添加新属性。

10.1.3. 本地目录树设计

根据前面部分描述的数据和模式设计， Example Corp. 创建以下目录树：

- 目录树的根是 Example Corp.'s Internet 域名： dc=example,dc=com。
- 目录树有四个分支点： ou=people,ou=groups,ou=roles, 和 ou=resources。
- 所有 Example Corp. 的人条目都在 ou=people 分支下创建。

  人员条目是 个人、organizationalPerson、inetOrgPerson 和 examplePerson 对象类的所有成员。uid 属性唯一标识每个条目的 DN。例如， Example Corp. 包含 Babs Jensen (uid=bjensen) 和 Emily Stanton (uid=estanton)的条目。
- 他们创建了三个角色，为示例企业中的每个部门创建一个角色：销售、营销和会计。

  每个人条目都包含一个 role 属性，用于标识该人员所属的部门。example Corp. 现在可以根据这些角色创建 ACI。

  有关角色的更多信息，请参阅 第 4.3.2 节 “关于角色”。

- 它们在 ou=groups 分支下创建两个组分支。

  第一个组 cn=administrators 包含管理目录内容的目录管理员的条目。

  第二个组 cn=messaging admin 含有管理邮件帐户的邮件管理员的条目。此组对应于 messaging 服务器使用的管理员组。示例 Corp. 确保它为消息传递服务器配置的组与为 Directory 服务器创建的组不同。

- 它们在 ou=resources 分支下创建两个分支，一个用于会议房间(ou=conference rooms)，另一个用于办公室(ou=offices)。

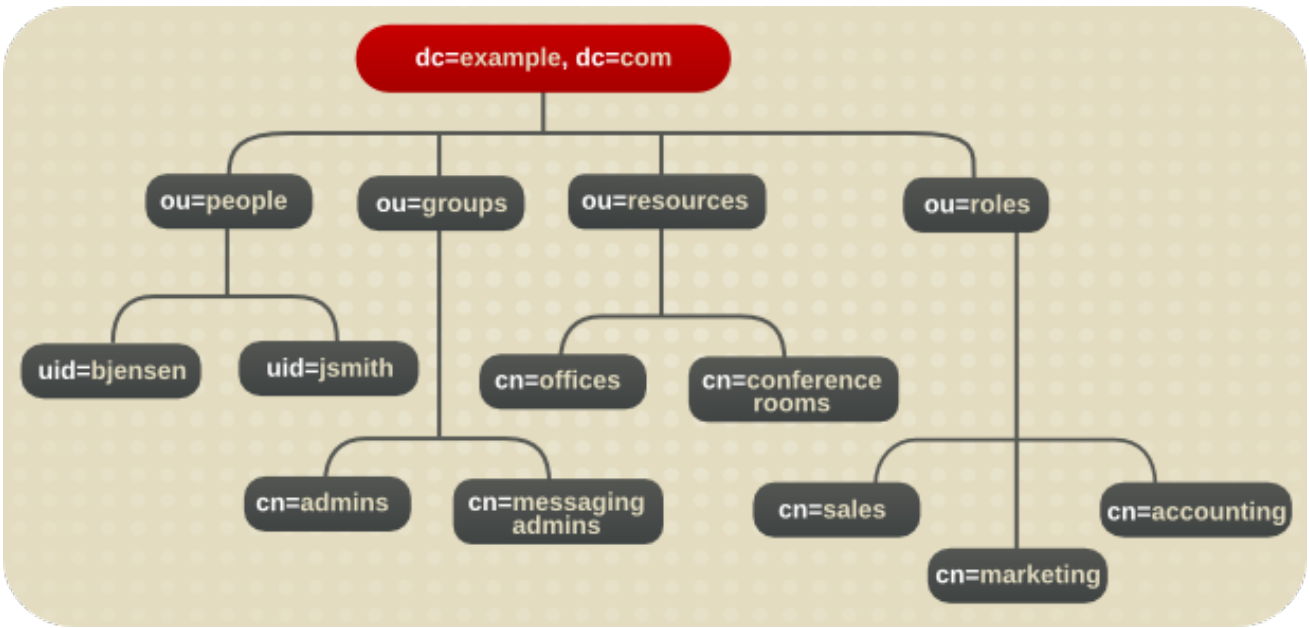
- 它们创建 类服务(CoS)，根据条目是否属于管理组，为 mailquota 属性提供值。

  这个 CoS 为管理员提供了 100GB 的邮件配额，而普通示例公司。员工拥有 5GB 邮件配额。

  有关服务类的更多信息，请参阅 第 5.3 节 “关于服务类”。

下图显示了从上面列出的设计步骤生成的目录树：

图 10.1. 示例公司的目录树.



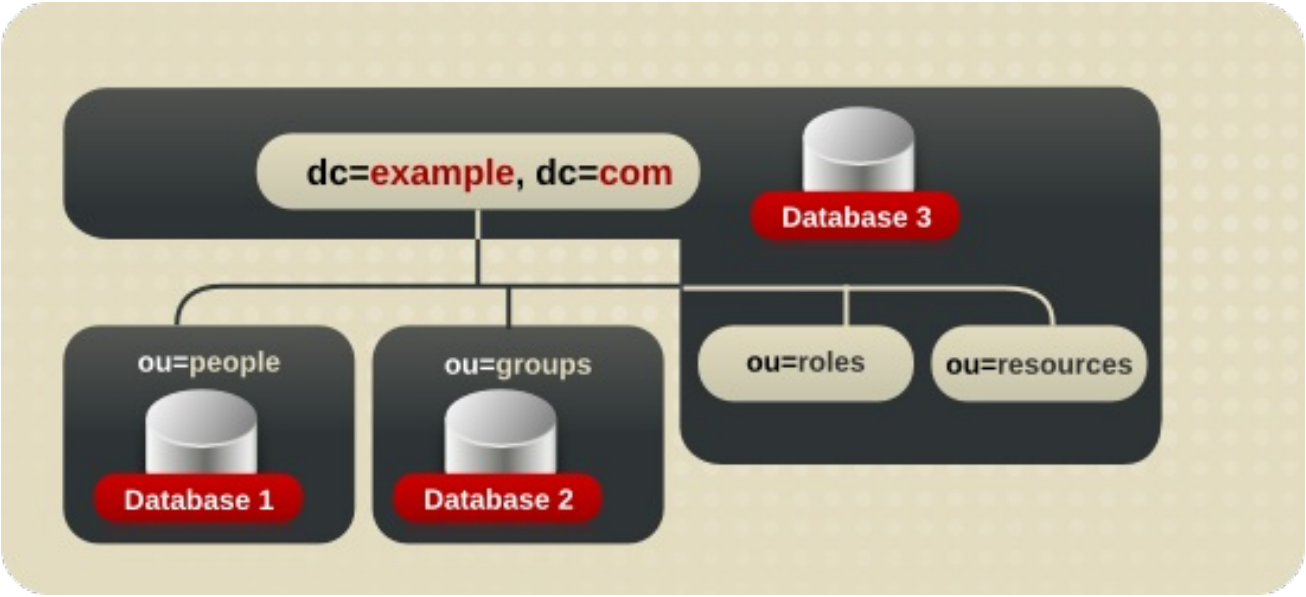
10.1.4. 本地 Enterprise Topology 设计

此时，Example Corp. 需要设计其数据库和服务器拓扑。以下小节详细介绍了每个拓扑。

10.1.4.1. 数据库拓扑

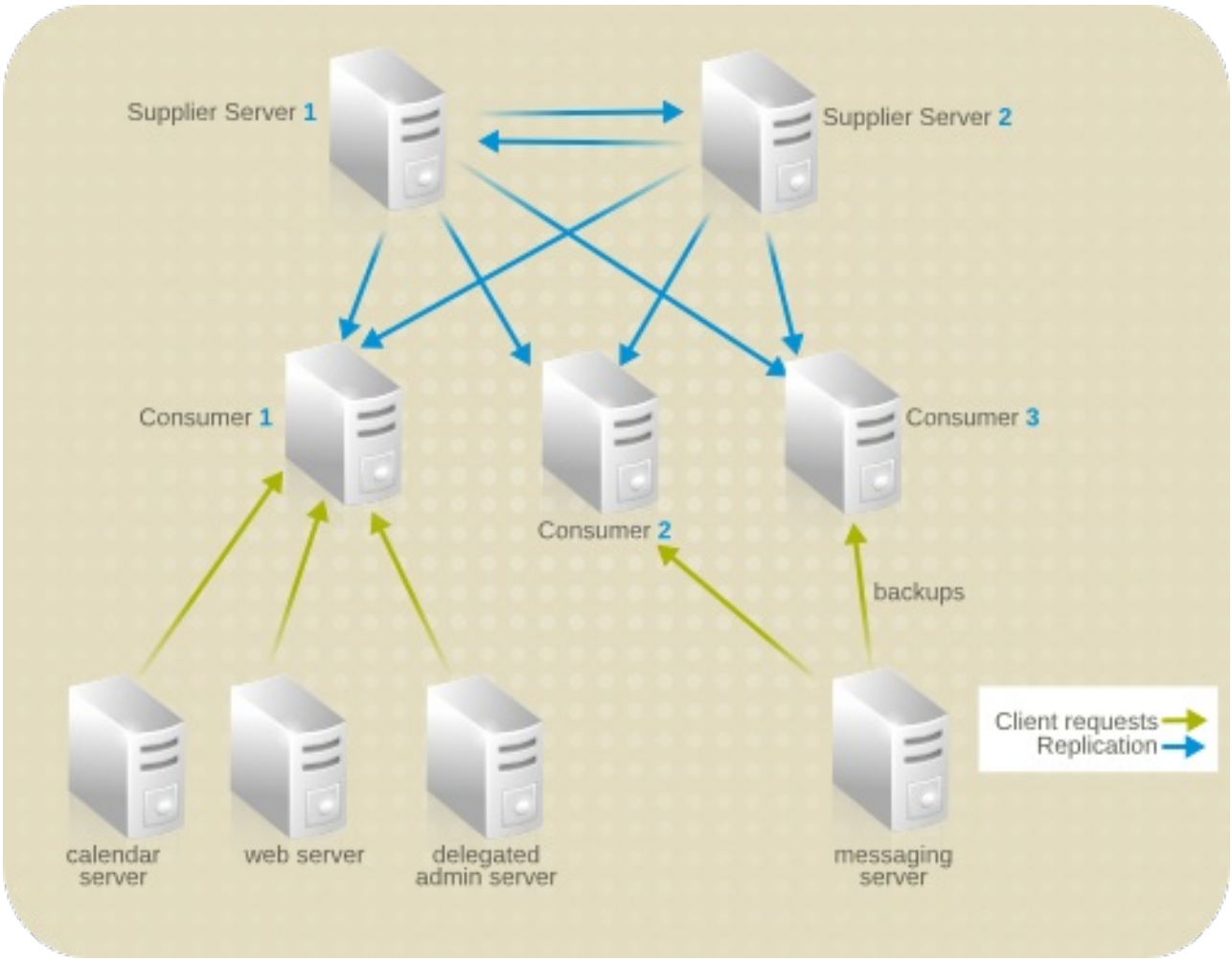
公司设计了一个数据库拓扑，其中人员分支存储在一个数据库(DB1)中，组分支存储在另一个数据库(DB2)中，资源分支、角色分支和 根后缀 信息存储在第三个数据库(DB3)中。图 10.2 “Corp 的数据库拓扑.” 中演示了这一点。

图 10.2. Corp 的数据库拓扑.



每个供应商服务器都会更新 Example Corp. 部署 Directory Server 中的所有三个消费者服务器。这些消费者向一个消息传递服务器以及其它统一的用户管理产品提供数据。

图 10.3. 示例公司的服务器拓扑.



修改 来自兼容服务器的请求将路由到适当的消费者服务器。使用者服务器使用智能引用将请求路由到负责修改数据的主副本的供应商服务器。

10.1.5. 本地企业复制设计

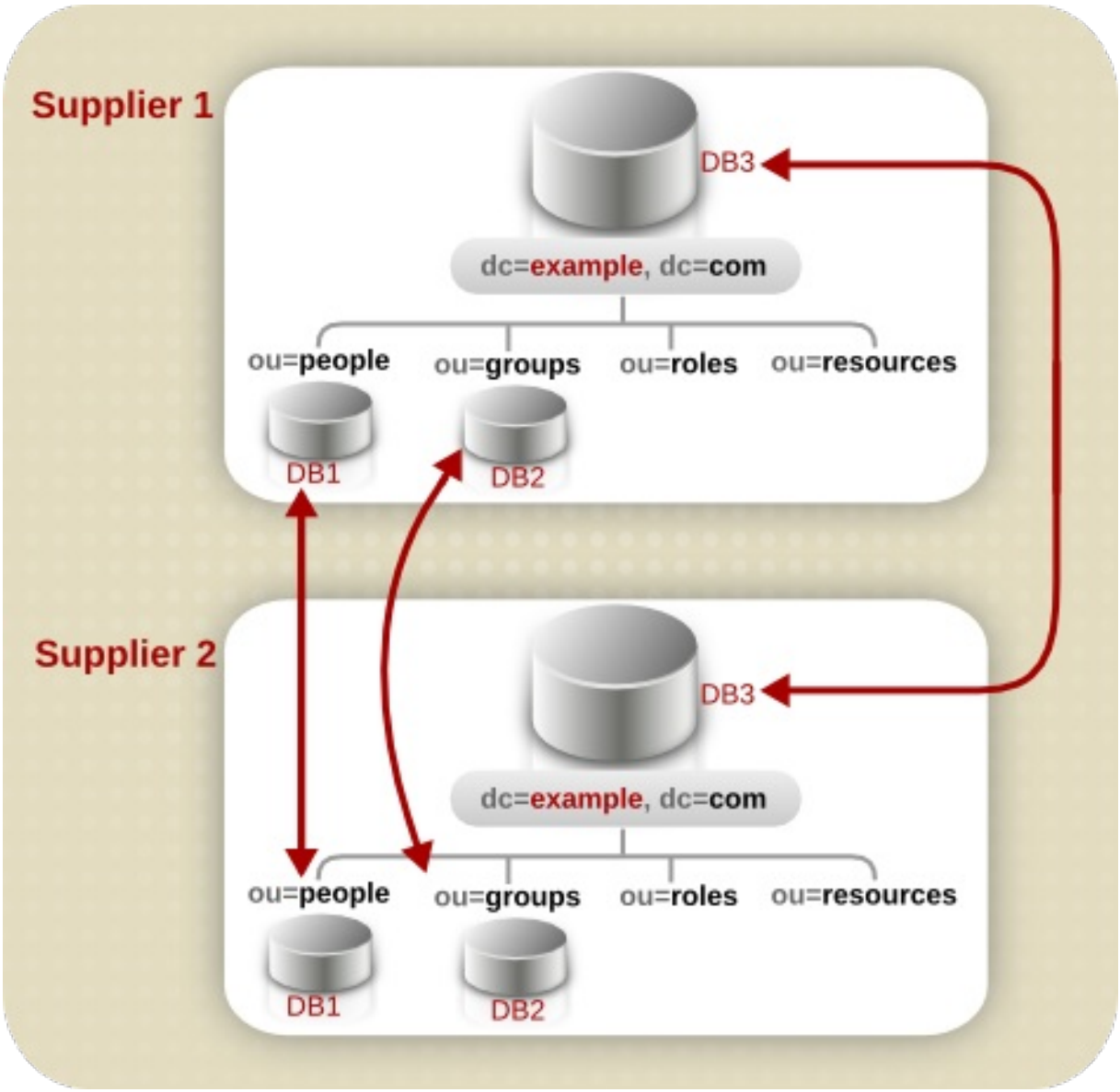
示例. 决定使用多层次复制设计来确保其目录数据的高可用性。有关多倍复制的更多信息，请参阅 第 7.2.2 节 "Multi-Supplier Replication"。

以下小节提供有关供应商服务器架构和 Vendor-consumer 服务器拓扑的更多详细信息。

10.1.5.1. 供应商架构

示例公司在多路复制架构中使用两个供应商服务器。供应商更新另一个目录数据，以便目录数据保持一致。示例 Corp. 的供应商架构如下所述：

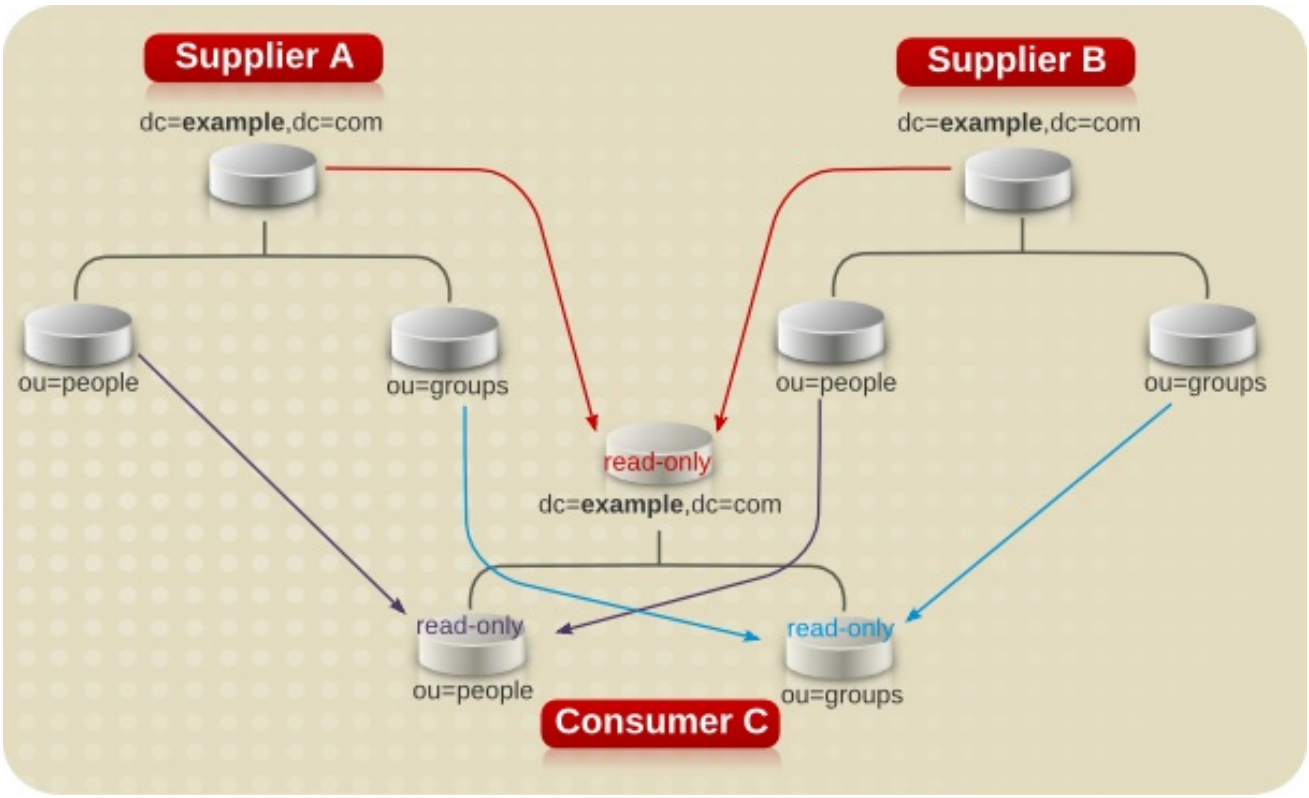
图 10.4. 企业供应商架构示例.



10.1.5.2. 供应商消费者架构

下图显示了供应商服务器如何将供应商服务器复制到 Example Corp. 部署 目录中的每个消费者。三个消费者服务器各自由两个供应商服务器更新。这样可确保在供应商服务器中有故障时用户不会受到这个安全漏洞的影响。

图 10.5. 企业示例供应商和消费者架构.



10.1.6. 本地企业安全设计

Corp. 示例决定以下安全设计来保护其目录数据：

- 它们创建一个 ACI，使员工能够修改自己的条目。

  用户可以修改除 uid、管理器 和部门属性 以外的所有 属性。

- 为了保护员工数据的隐私，他们会创建一个 ACI，仅允许员工及其经理查看员工的主页地址和电话号码。

- 它们在目录树的根目录处创建一个 ACI，允许两个管理员对适当的目录权限进行分组。

  目录管理员组需要对该目录具有完全访问权限。消息传递管理员组需要对 mailRecipient 和 mailGroup 对象类以及 mailGroup 对象类以及 mail 属性的访问。示例 Corp. 还授予消息传递管理员组 write、delete，并将 权限添加到组子目录以创建邮件组。

- 它们在目录树的根目录处创建常规 ACI，允许匿名访问读取、搜索和比较访问。

这个 ACI 拒绝对密码信息进行匿名写入访问。

- 为了防止服务器拒绝服务攻击和不当使用，它们会根据要绑定的目录客户端的 DN 设置资源限值。

  示例 Corp. 允许匿名用户在响应搜索请求时收到 100 个条目，以响应搜索请求、消息管理用户接收 1,000 个条目，以及目录管理员获得无限个条目。

  有关根据绑定 DN 设置资源限值的更多信息，请参阅 红帽目录服务器管理员指南中的"用户帐户管理"一章。

- 它们会创建一个密码策略，指定密码必须至少为 8 个字符，并在 90 天后过期。

  有关密码策略的更多信息，请参阅 第 9.6 节 “设计密码策略”。

- 它们创建一个 ACI，为财务角色成员提供对所有付费信息的访问权限。

10.1.7. 本地企业操作决策

公司对其目录的日常运作做出以下决策：

- 每晚备份数据库。
- 使用 SNMP 监控服务器状态。

  有关 SNMP 的更多信息，请参阅 红帽目录服务器管理员指南。

- 自动轮转访问和错误日志。
- 监控错误日志，以确保服务器按预期执行。

- 监控访问日志到屏幕，以查看应索引的搜索。

有关访问、错误和审计日志的更多信息，请参阅 红帽目录服务器管理员指南中的"监控服务器和数据库活动"一章。

10.2. 设计示例：多企业及其 EXTRANET

本例为 example Corp 构建了一个目录基础架构。国际.上一个示例中的 example Corp. 增长为一个大型的跨国公司。本示例基于示例 Corp. 中创建的目录结构，扩展目录设计以满足其新需求。

示例公司已发展成为一个组织分布在三个主要地理位置：美国、欧洲和亚洲。示例. 现在，公司拥有 20,000 多个员工，在 Example Corp. 办公室的国家/地区实时和工作。示例. 决定启动公司范围的 LDAP 目录以改进内部通信，以便更轻松地开发和部署 Web 应用程序，从而提高安全性和隐私。

为国际企业设计目录树涉及确定如何以逻辑方式收集目录条目、如何支持数据管理以及如何支持全局规模上的复制。

此外，示例企业还希望创建一个额外的网来供其部分供应商和交易合作伙伴使用。extranet 是企业内部网到外部客户端的一个扩展。

以下小节描述了部署多证书目录服务以及 Example Corp 的 extranet 过程中的步骤。国际.

10.2.1. 跨性企业数据设计

企业示例.国际创建部署团队来执行站点调查。部署团队从站点问卷调查中确定以下内容：

- 消息传递服务器用于为大多数 Example Corp. 站点提供电子邮件路由、交付和读取服务。企业服务器提供文档发布服务。所有服务器均在 Red Hat Enterprise Linux 7 上运行。

- 示例 Corp. 需要允许本地管理数据。例如，欧洲站点负责管理目录的欧洲分支。这也意味着，欧洲负责其数据的主副本。

- 由于 Example Corp. 的办公室的地理分布，因此目录需要每天 24 小时提供给用户和应用程序。

- 许多数据元素需要容纳多种不同语言的数据值。

注意

所有数据都使用 UTF-8 字符集；任何其他字符集都违反了 LDAP 标准。

部署团队还决定以下有关 extranet 的数据设计：

- 部分供应商需要登录到示例 Corp. 的 目录，以便管理其有 Example Corp 的合同。部分供应商依赖于用于身份验证的数据元素，如名称和用户密码。

- 示例企业的合作伙伴将使用目录查找合作伙伴网络中人员的联系详情，如电子邮件地址和电话号码。

10.2.2. 跨性企业架构设计

Corp. 示例通过添加 schema 元素来支持 extranet，以此构建其原始模式设计。示例 Corp. 添加两个新对象，即 exampleSupplier 对象类和 examplePartner 对象类。

示例供应商 对象类允许一个属性，即 exampleSupplierID 属性。此属性包含由 Example Corp 分配的唯一 ID。国际每个自治区部分供应商。

examplePartner 对象类允许一个属性 examplePartnerID 属性。此属性包含由 Example Corp 分配的唯一 ID。国际每个交易合作伙伴。

有关自定义默认目录模式的详情，请参考 第 3.4 节 “自定义架构”。

10.2.3. 多企业目录树设计

根据扩展的要求，Example Corp. 会创建以下目录树：

- 目录树的根目录是 dc=com 后缀。在此后缀下，Example Corp. 创建两个分支。一个分支(dc=exampleCorp,dc=com)，包含 Example Corp 的内部数据。国际.另一个分支

dc=exampleNet,dc=com 包含 extranet 的数据。

- *intranet 的目录树（在 dc=exampleCorp,dc=com） 有三个主要分支，各自对应于 Example Corp. 具有办事处的其中一个区域。这些分支使用 l （本地性）属性来标识。*

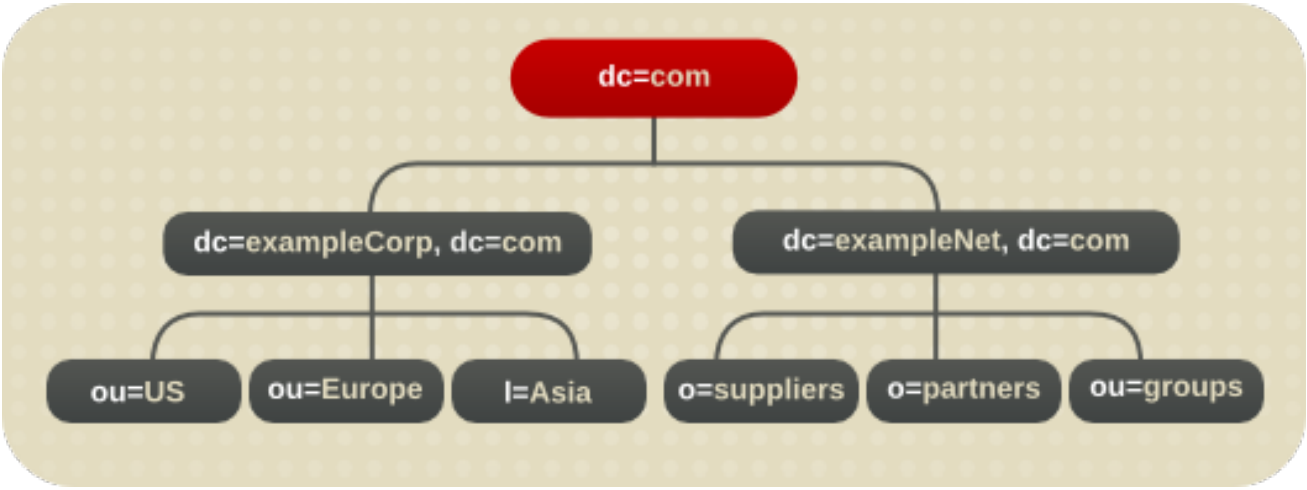
- *dc=exampleCorp,dc=com 下的每个主要分支模拟 Example Corp 的原始目录树设计。在每个地区下，Example Corp. 创建一个 ou=people、ou=groups、ou=roles 和 ou=resources 分支。有关这个目录树设计的更多信息，请参阅 图 10.1 “示例公司的目录树.”。*

- *在 dc=exampleNet,dc=com 分支下，Example Corp. 创建三个分支。供应商的一个分支 (o=suppliers)、合作伙伴(o= partners)的一个分支，另一个用于组(ou=groups)。*

- *extranet 的 ou=groups 分支包含 extranet 管理员的条目，以及用于邮件列表，供合作伙伴订阅有关 automobile 部分 manufacturing 中的最新信息。*

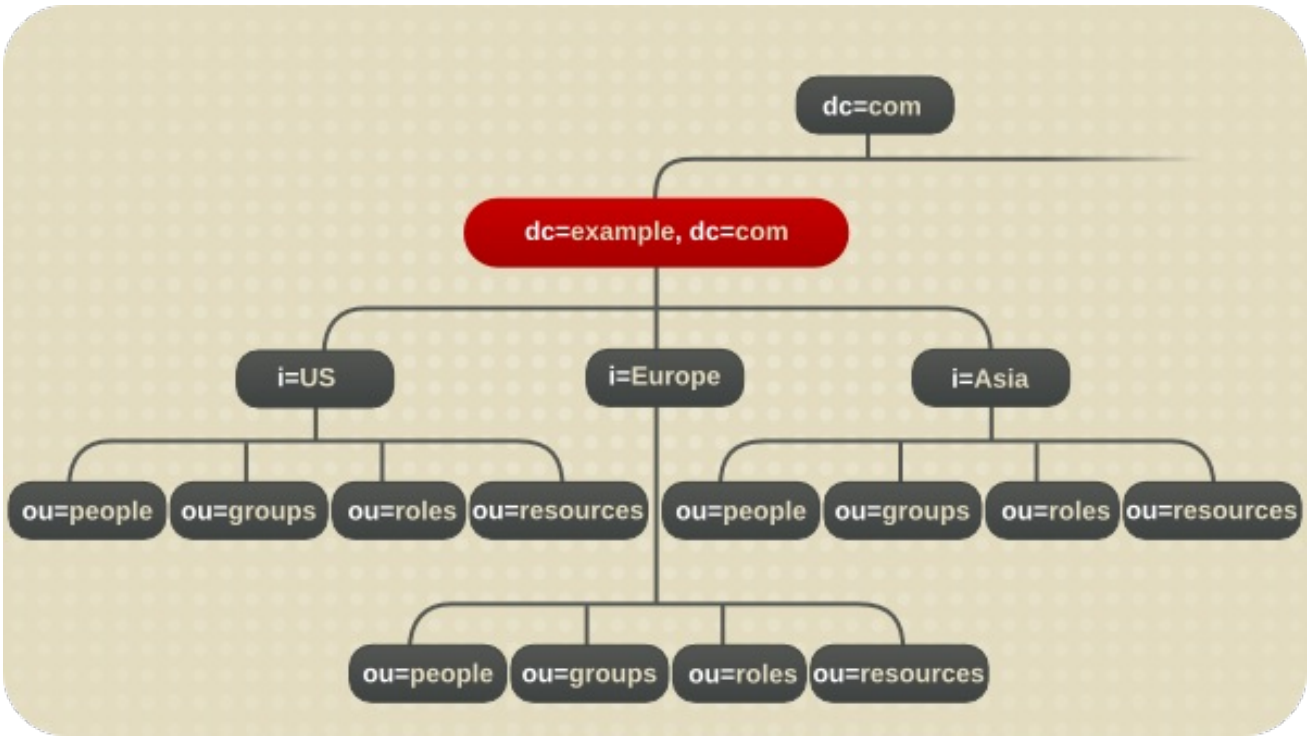
下图显示了从上面列出的设计步骤生成的基本目录树：

图 10.6. 示例公司的基本目录树.国际



下图演示了 Example Corp. intranet 的目录树：

图 10.7. 示例公司的目录树.国际 Intranet

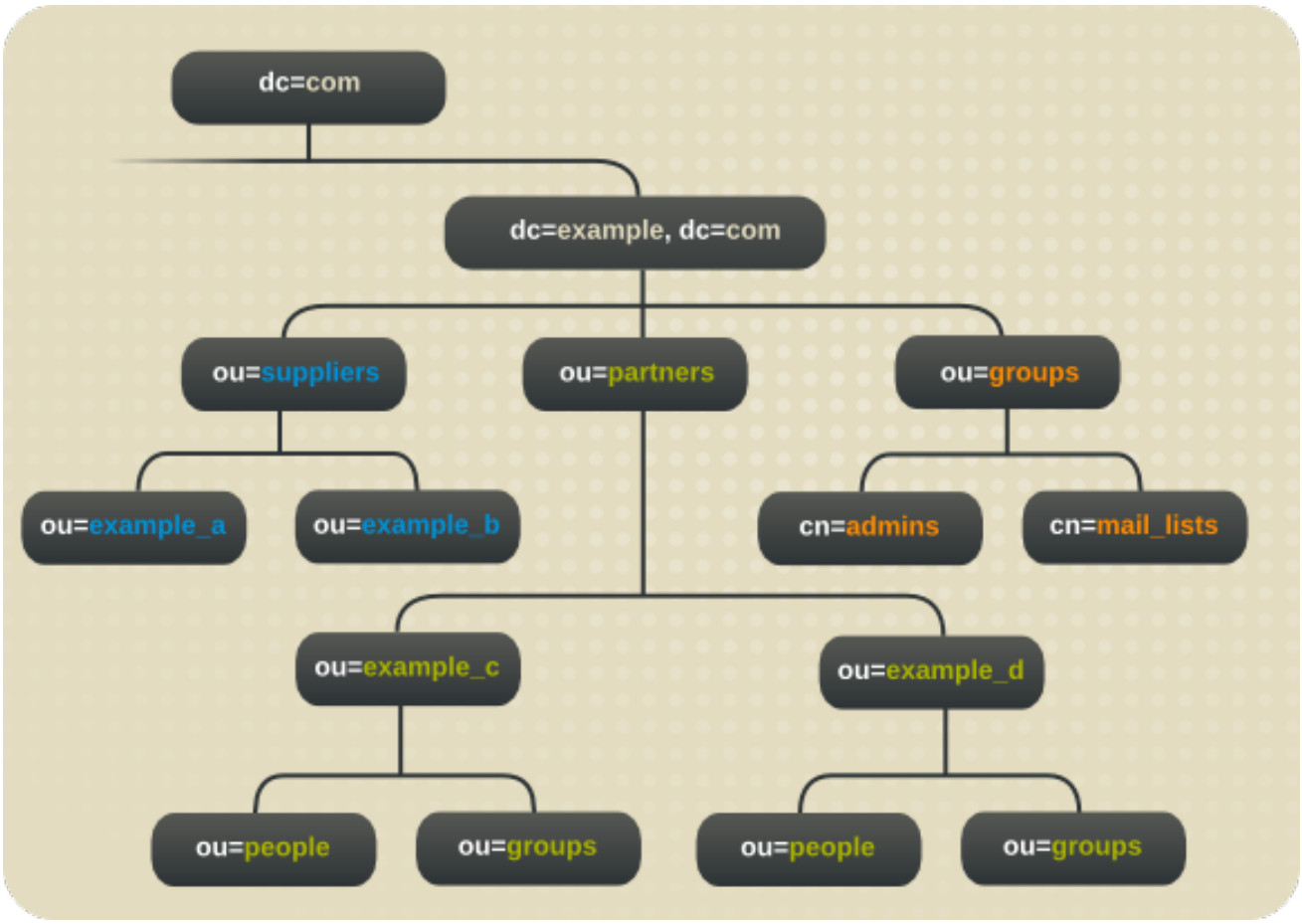


l=Asia 条目的条目会出现在 LDIF 中，如下所示：

```
dn: l=Asia,dc=exampleCorp,dc=com

objectclass: top

objectclass: locality

l: Asia

description: includes all sites in Asia
```

下图显示了 Corp. 的 extranet 的目录树：

图 10.8. 示例公司的目录树.国际 Extranet



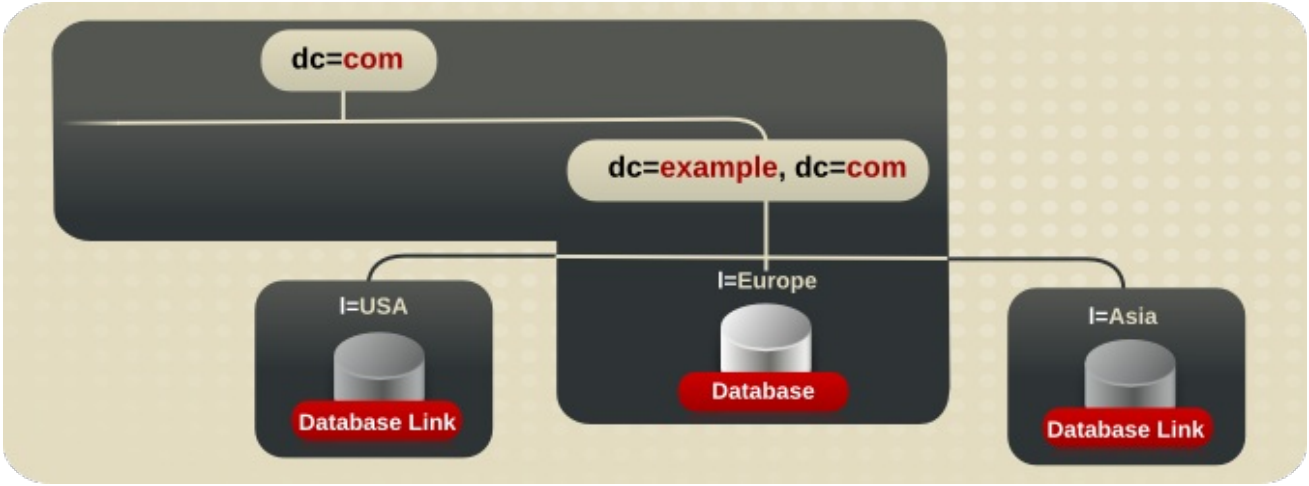
10.2.4. 跨性企业拓扑设计

此时，Example Corp. 会设计其数据库和服务器拓扑。以下小节更详细地描述了每个拓扑。

10.2.4.1. 数据库拓扑

下图演示了一个示例企业机构（欧洲、欧洲）的数据库拓扑：

图 10.9. Corp 的数据库拓扑.欧洲



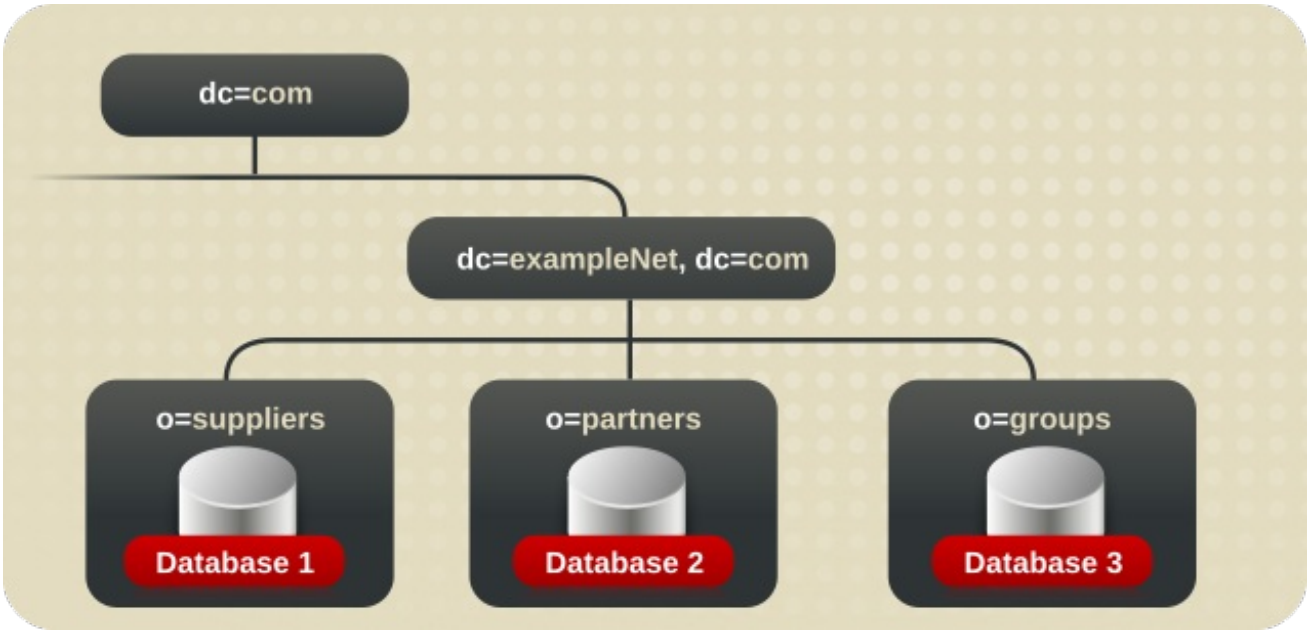
数据库链接指向存储在每个国家/地区的数据库。例如，由 Example Corp 接收的操作请求。l=US 分支下数据的欧洲服务器由 Austin, Texas 的服务器上的数据库链接串联。有关数据库链接和链的详情，请参考 第 6.3.2 节 “使用链”。

dc=exampleCorp,dc=com 和 root 条目 dc=com 的数据的主副本存储在 l=Europe 数据库中。

欧洲的数据中心包含 extranet 的数据的主副本。extranet 数据存储在三个数据库中，每个主分支对应一个。o=suppliers 数据的主副本存储在数据库 1 (DB1)中，针对 o= partners 存储在数据库 2 (DB2)中，而 ou=groups 则存储在数据库三(DB3)中。

extranet 的数据库拓扑如下所示：

图 10.10. Corp 的数据库拓扑.国际 Extranet



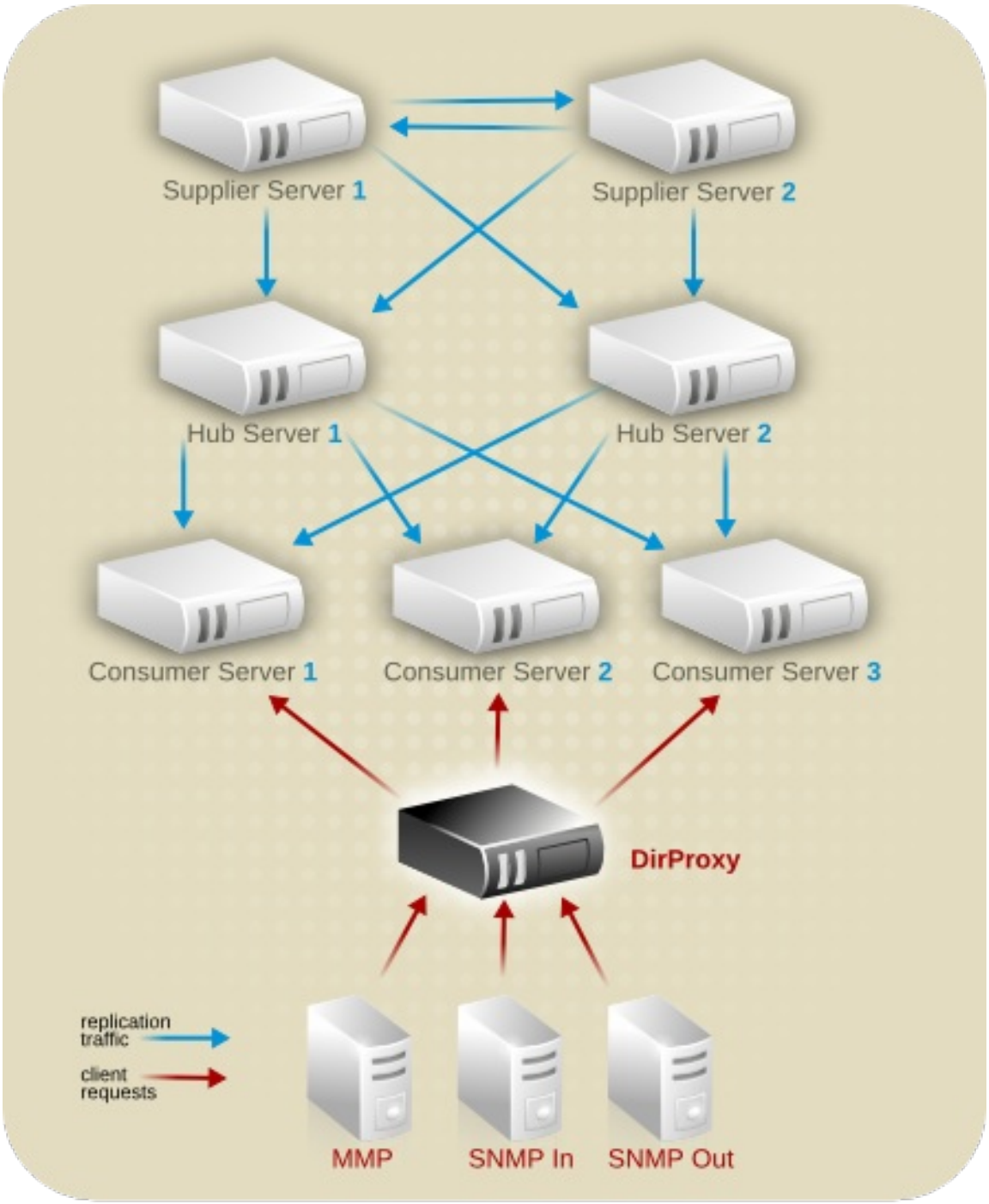
10.2.4.2. 服务器拓扑

示例公司开发两个服务器拓扑，一个用于公司内部，另一个用于合作伙伴的 extranet。

对于内部网，example Corp. 决定为每个主要地点有供应商数据库。这意味着，它有三个数据中心，分别包含两个供应商服务器、两个 hub 服务器和三个消费者服务器。

下图演示了公司示例企业的架构。欧洲的数据中心：

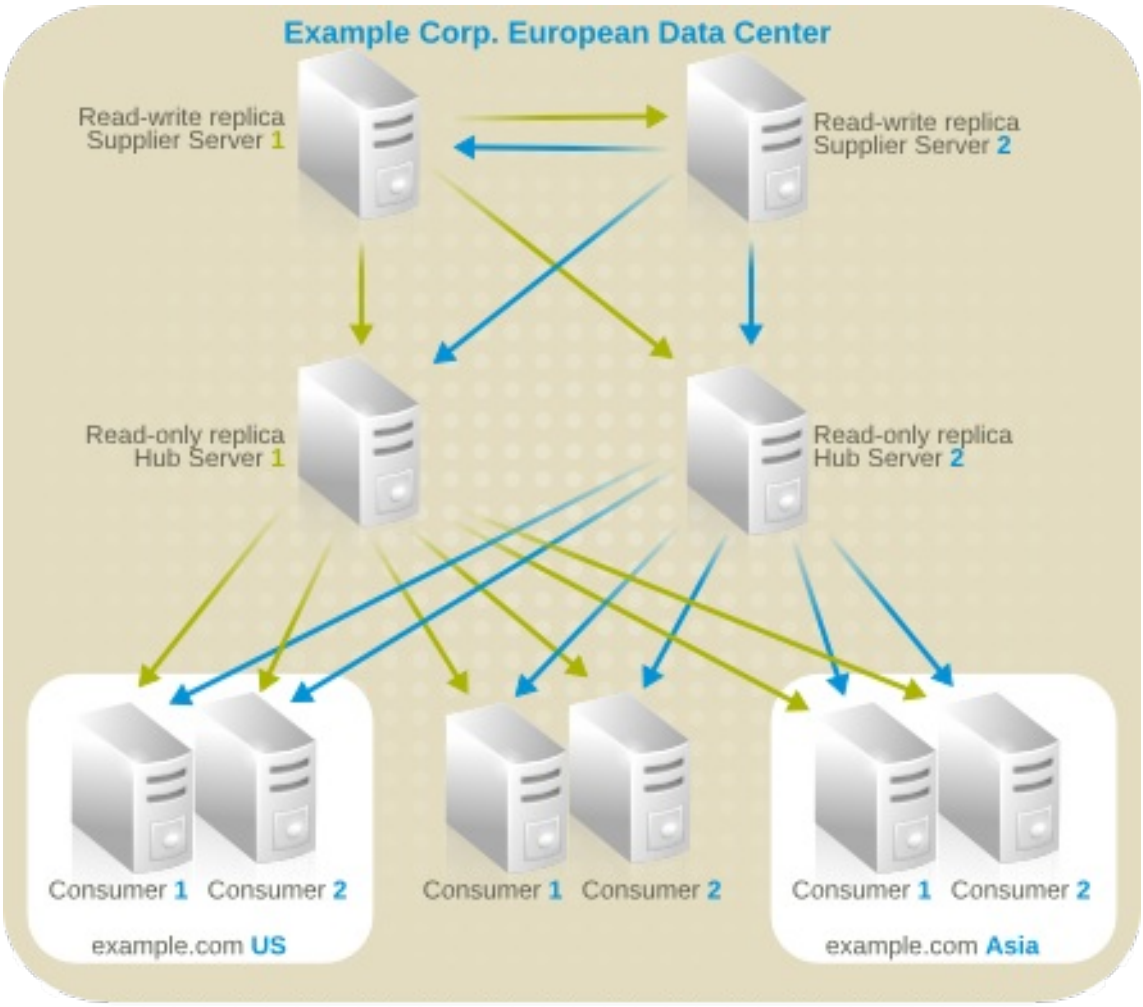
图 10.11. 示例公司的服务器拓扑.欧洲



示例公司的 extranet 的数据供应商位于欧洲。此数据被复制到美国数据中心的两台消费者服务器，以及 Asia 数据中心中的两个消费者服务器。总体而言，example Corp. 需要十个服务器来支持 extranet。

下图演示了 European 数据中心中 Example Corp. 的 extranet 的服务器架构：

图 10.12. 示例公司的服务器拓扑.国际 Extranet



hub 服务器将数据复制到欧洲、美国和亚洲每个数据中心的两个消费者服务器。

10.2.5. 跨性企业复制设计

example Corp. 在为其目录设计复制时会考虑以下几点：

- *数据将在本地管理。*
- *网络连接质量因站点而异。*
- *数据库链接将用于连接远程服务器上的数据。*
- *包含数据的只读副本的 hub 服务器将用于复制数据到消费者服务器。*

中心服务器位于几乎重要的目录应用程序，如邮件服务器或 Web 服务器。

中心服务器从供应商服务器中移除复制的负担，因此供应商服务器可以专注于写操作。未来，随着 Example Corp. 扩展并需要添加更多的消费者，额外的消费者不会影响供应商服务器的性能。

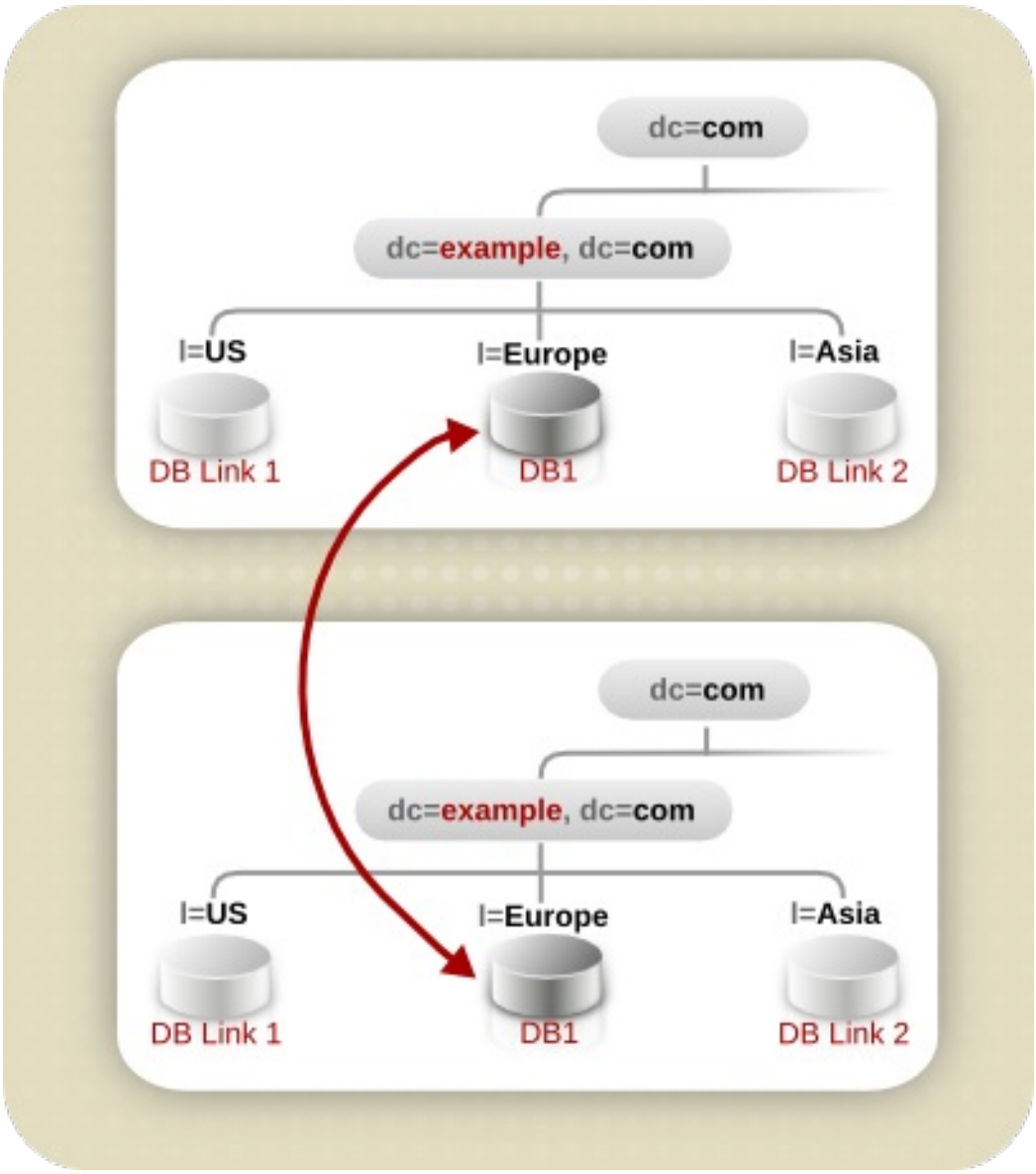
有关 hub 服务器的详情，请参考 第 7.2.3 节 “cascading Replication”。

10.2.5.1. 供应商架构

对于 Example Corp. intranet，每个本地化内容都会保存其数据的主副本，并使用数据库链接到其他地方的数据。对于其数据的主副本，每个本地化均使用多层次复制架构。

下图演示了欧洲的供应商架构，其中包括 dc=exampleCorp,dc=com 和 dc=com 信息：

图 10.13. 企业供应商架构示例.欧洲

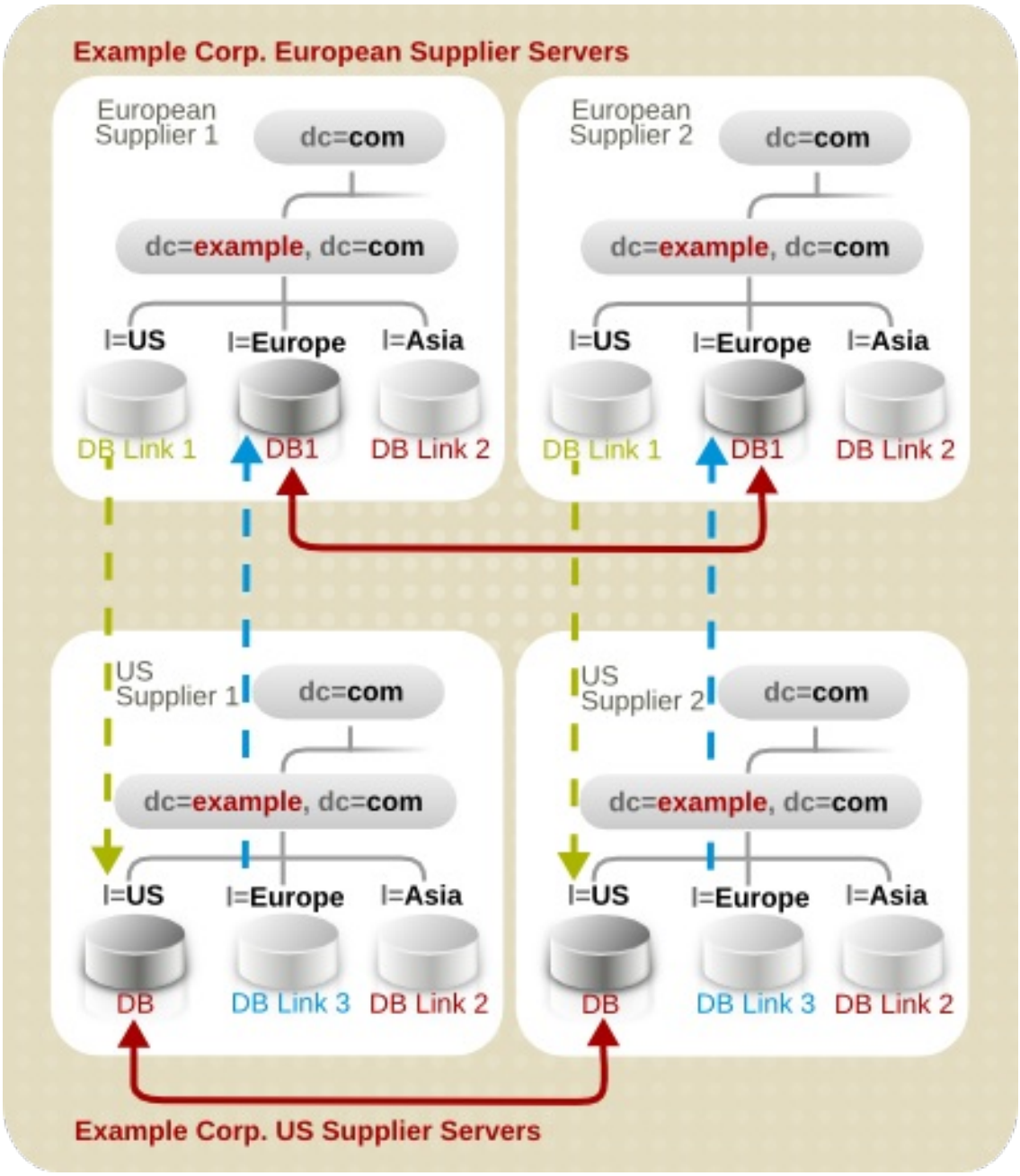


每个本地化内容包括两个供应商，它们共享该站点的数据的主副本。因此，每个位置都负责自己的数据的主副本。使用多supplier 架构可确保数据的可用性，并帮助平衡每个供应商服务器管理的工作负载。

为降低总体故障风险， Acamic. 在每个站点使用多个读写供应商目录服务器。

下图演示了欧洲两个供应商服务器与美国两个供应商服务器之间的互动：

图 10.14. 示例公司的多层次复制设计.欧洲和示例公司.US



示例公司之间存在同样的关系。美国和示例公司.亚洲，以及企业示例之间的.欧洲和示例公司.亚洲.

10.2.6. 跨性企业安全设计

企业示例.国际构建之前的安全设计，添加以下访问控制来支持其新的跨国内部网：

- *示例 Corp. 将常规 ACI 添加到内部的根目录中，在每个国家/地区中创建更严格的 ACI，以及各个国家下的分支。*

- *Corp. 决定使用宏 ACI 来最小化目录中的 ACI 数量。*

  示例 Corp. 使用宏代表 ACI 目标中的 DN 或绑定规则部分。当目录获得传入 LDAP 操作时，ACI 宏与 LDAP 操作目标的资源匹配。如果存在匹配项，则宏替换为目标资源的 DN 值。

  有关宏 ACI 的更多信息，请参阅 红帽目录服务器管理员指南。

示例。添加了以下访问控制来支持其 extranet：

- *Corp. 决定对所有 extranet 活动使用基于证书的身份验证。当人们登录到 extranet 时，他们需要一个数字证书。目录用于存储证书。由于 目录存储了证书，因此用户可以通过查找保存在目录中的公钥来发送加密的电子邮件。*

- *示例 Corp. 创建一个 ACI，用于禁止对 extranet 的匿名访问。这样可防止 extranet 拒绝服务攻击。*

- *示例 Corp. 希望更新目录数据，使其仅来自 example Corp. 托管的应用程序。这意味着，使用 extranet 的合作伙伴和供应商只能使用 Example Corp 提供的工具。将 extranet 用户限制为 Example Corp. 的首选工具允许 example Corp. 管理员使用审计日志跟踪目录的使用，并限制 Example Corp 外部的 extranet 用户可以引入的问题类型。国际.*

附录 A. 目录服务器 RFC 支持

注意

本章列出了支持的 LDAP 相关 RFC。它不是 RFCs Directory Server 支持的完整列表。

A.1. LDAPV3 功能

技术规范路线图(RFC 4510)

这是一个跟踪文档，不包含要求。

协议(RFC 4511)

支持。例外：

- RFC 4511 第 4.4.1 节.请注意 Disconnection ：Directory 服务器在这种情况下终止连接。
- RFC 4511 第 4.5.1.3 节：SearchRequest.derefAliases 不支持 LDAP 别名。
- RFC 4511 第 4.13.IntermediateResponse 消息

目录信息模型(RFC 4512)

支持。例外：

- RFC 4512 第 2.4.2 节.结构化对象类 ：目录服务器支持多个结构对象类的条目。
- RFC 4512 第 2.6 节.别名条目
- RFC 4512 第 4.1.2 节：attribute Types ：不支持属性类型 COLLECTIVE。

- RFC 4512 第 4.1.4 节：匹配规则使用
- RFC 4512 第 4.1.6 节.DIT 内容规则
- RFC 4512 第 4.1.7 节：DIT 结构规则和名称表单
- RFC 4512 Section 5.1.1. altServer

请注意，RFC 4512 启用 LDAP 服务器不支持之前列出的异常。详情请查看 RFC 4512 第 7.1 部分。服务器指南.

验证方法和安全性机制(RFC 4513)

支持。

可辨识名称的字符串(RFC 4514)

支持。

搜索过滤器的字符串(RFC 4515)

支持。

统一资源查找器(RFC 4516)

支持。但是，这个 RFC 主要侧重于 LDAP 客户端。

语法和匹配规则(RFC 4517)

支持。例外：

- directoryStringFirstComponentMatch
- integerFirstComponentMatch

- *objectIdentifierFirstComponentMatch*
- *objectIdentifierFirstComponentMatch*
- *keywordMatch*
- *wordMatch*

国际化的字符串准备(RFC 4518)

支持。

用户应用的模式(RFC 4519)

支持。

A.2. 验证方法

匿名 SASL 机制(RFC 4505)

不支持。请注意，RFC 4512 不需要 ANONYMOUS SASL 机制。但是，Directory 服务器支持 LDAP 匿名绑定。

外部 SASL 机制(RFC 4422)

支持。

纯 SASL 机制(RFC 4616)

不支持。请注意，RFC 4512 不需要 PLAIN SASL 机制。但是，Directory 服务器支持 LDAP 匿名绑定。

SecurID SASL Mechanism(RFC 2808)

不支持。但是，如果出现 Cyrus SASL 插件，Directory 服务器就可以使用它。

Kerberos V5(GSSAPI)SASL 机制(RFC 4752)

支持。

CRAM-MD5 SASL 机制(RFC 2195)

支持。

摘要-MD5 SASL 机制(RFC 2831)

支持。

一次性密码 SASL 机制(RFC 2444)

不支持。但是，如果出现 Cyrus SASL 插件，Directory 服务器就可以使用它。

A.3. X.509 证书架构和属性支持

X.509 证书的 LDAP 架构定义(RFC 4523)

- 属性类型和对象类：支持的.
- 语法：不支持。目录服务器使用二进制和八位字节语法。
- 匹配规则：不支持。

附录 B. 修订历史记录

请注意，修订号与该手册版本相关，而不是与 Red Hat Directory Server 的版本号相关。

修订 11.5-1 **Tue May 10 2022** **Marc Muehlfeld**

本指南的 Red Hat Directory Server 11.5 发行版本。

修订 11.4-1 **Tue Nov 09 2021** **Marc Muehlfeld**

本指南的 Red Hat Directory Server 11.4 发行版本。

修订 11.3-1 **Tue May 11 2021** **Marc Muehlfeld**

本指南的 Red Hat Directory Server 11.3 发行版本。

修订 11.2-1 **Tue Nov 03 2020** **Marc Muehlfeld**

本指南的 Red Hat Directory Server 11.2 发行版本。

修订 11.1-1 **Tue Apr 28 2020** **Marc Muehlfeld**

本指南的 Red Hat Directory Server 11.1 发行版本。

修订 11.0-1 **Thu Nov 21 2019** **Marc Muehlfeld**

本指南的 Red Hat Directory Server 11.0 发行版本。