**LDAP обзор и понятия (вот так бы все объясняли)**

Плюсы:

* Он стандартизирован, поэтому довольно часто уже существуют некоторые обязательные плагины;
* Многие фреймворки разработки поддерживают связь с LDAP; (Я сделал это с PHP некоторое время назад; у .NET есть специальное пространство имен для этого.)
* Существуют все необходимые методы аутентификации и механизмы безопасности-пароли хранятся надежно, аутентификация с сервером может быть выполнена защищенным способом, если это необходимо;
* При необходимости можно включить функции политики блокировки учетной записи/истории паролей;
* AD имеет удобные инструменты для управления пользователями, также он может быть написан по сценарию или доступен через API;
* Пользователи могут иметь сложное членство в группах;
* Разрешения на любой объект/атрибут можно настроить;
* Каталог может быть распространен/реплицирован путем добавления дополнительных контроллеров домена.

Аферы:

* Deployment требует некоторого планирования;
* ActiveDirectory следует регулярно обновлять.
* Если вы аутентифицируетесь в Active Directory, вам нужны лицензии Microsoft.
* Лицензия стоит довольно дорого.

LDAP и X.500 имеют много общих терминов, некоторые из них важны, некоторые — просто ерунда. Для Вашего удобства мы создали [глоссарий](https://pro-ldap.ru/tr/zytrax/apd). Термины в него включены либо потому, что они важны, либо потому, что они часто используются в литературе.

[2.1 Краткая история LDAP](https://pro-ldap.ru/tr/zytrax/ch2/#history)  
[2.2 Обзор LDAP](https://pro-ldap.ru/tr/zytrax/ch2/#overview)  
[2.3 LDAP и базы данных](https://pro-ldap.ru/tr/zytrax/ch2/#database)  
[2.3.1 Использование LDAP — резюме](https://pro-ldap.ru/tr/zytrax/ch2/#summary)  
[2.4 Модель данных (объектная модель) LDAP](https://pro-ldap.ru/tr/zytrax/ch2/#model)

[2.4.1 Структура дерева объектов](https://pro-ldap.ru/tr/zytrax/ch2/#basic)  
[2.4.2 Объектные классы](https://pro-ldap.ru/tr/zytrax/ch2/#objectclasses)  
[2.4.3 Атрибуты](https://pro-ldap.ru/tr/zytrax/ch2/#attributes)  
[2.4.4 Описание дерева путём добавление записей (данных)](https://pro-ldap.ru/tr/zytrax/ch2/#data)  
[2.4.5 Навигация по дереву](https://pro-ldap.ru/tr/zytrax/ch2/#navigation)

[2.5 Отсылки и репликация LDAP](https://pro-ldap.ru/tr/zytrax/ch2/#organization)

[2.5.1 Отсылки](https://pro-ldap.ru/tr/zytrax/ch2/#referrals)  
[2.5.2 Репликация](https://pro-ldap.ru/tr/zytrax/ch2/#replication)

**Небольшое замечание о чувствительности к регистру в LDAP:** здесь есть некоторая путаница, по крайней мере, мы считаем это путаницей. По правде говоря, путаниц тут хватает на каждом шагу. Единственные чувствительные к регистру вещи в LDAP — это пароли и содержимое **отдельных** (очень малоизвестных) атрибутов, в зависимости от их [правил соответствия](https://pro-ldap.ru/tr/zytrax/apd/index.html#matchingrule). Всё. В этой и других документациях Вы будете встречать и objectclass, и objectClass, и даже ObjectClass. Все эти формы работают. Точка. После первых шести лет изучения LDAP (шутка, конечно, хватит и четырёх) Вы уже не будете хвататься за сердце всякий раз, когда Вам доведётся опечататься в каком-нибудь имени. Итак, использование "верблюжьей" нотации — это, возможно, и неплохая практика, но даже если Вы где-то ошибётесь, солнце не упадёт на землю.

**О букве D в LDAP:** официально буква D в аббревиатуре LDAP означает Directory (каталог) — Lightweight Directory Access Protocol. Связано это главным образом с историческими истоками LDAP (и его предшественника DAP), которые были ориентированы на взаимодействие с классическими приложениями для работы с каталогом адресов электронной почты типа "белые страницы". Однако, любая терминология в конечном итоге может начать загонять саму себя и тех, кто ею пользуется, в какие-то рамки. Не заблуждайтесь, рассуждая о LDAP, мы говорим о доступе к **данным**, и если термин **Directory** ограничивает Ваше мышление из-за существующих ментальных моделей каталогов (наше уж точно ограничивает, хотя, возможно, мы просто сами по себе умственно ограничены), то, размышляя о LDAP, просто мысленно заменяйте его термином **Data** (данные), и получится Lightweight Data Access Protocol. Только никому не говорите, а то прослывёте вероотступником.

**2.1 Краткая история LDAP**

Когда-то в тёмном и далёком прошлом (конец 70-х — начало 80-х) [ITU (International Telecommunication Union)](http://www.itu.int/) начал работу над почтовыми стандартами серии X.400. Для этого стандарта требовался каталог имён (и другой информации), который мог бы быть доступен по сети в иерархической манере, схожей с DNS (для тех из Вас, кто знаком с её архитектурой).

Эта потребность в глобальном сетевом каталоге сподвигла ITU к разработке стандартов серии X.500 и, в частности X.519, которые определяют [DAP](https://pro-ldap.ru/tr/zytrax/apd/index.html#dap) (Directory Access Protocol), протокол для доступа к сетевой службе каталогов.

Серии стандартов X.400 и X.500 разрабатывались как составная часть полного стека OSI и были большими, громоздкими и потребляли много ресурсов. Фактически, стандартная ситуация для ITU.

Перенесёмся в начало 90-х. IETF осознала необходимость доступа к глобальным службам каталогов (первоначально, во многом, по тем же самым причинам хранения адресов электронной почты, что и ITU), но не поднимая при этом всех этих ужасных перенагруженных протоколов (OSI), и начала работу над Lightweight Directory Access Protocol (LDAP). LDAP разрабатывался так, чтобы обеспечить почти столько же функциональности, что и оригинальный стандарт X.519, но с использованием стека протоколов TCP/IP, при этом оставляя возможность взаимодействия с каталогами, основанными на X.500. Действительно, взаимодействие с X.500 (DAP) и его отображение всё ещё является частью серии RFC о LDAP от IETF.

Большинство вопросов в спецификациях LDAP, вызывающих серьёзную головную боль, как раз связаны с обратной совместимостью с X.500 и концепцией глобальной службы каталогов. Самый показательный из них — соглашение об [именовании корневой записи](https://pro-ldap.ru/tr/zytrax/apa/ldap-root.html).

В широком смысле, LDAP отличается от DAP в следующих аспектах:

1. В LDAP используется TCP/IP — DAP использует OSI в качестве транспортного/сетевого слоёв.
2. Некоторое сокращение функциональности — неясные, дублирующиеся и редко используемые функции X.519 (конёк ITU) тихо и благополучно отброшены.
3. Замена в LDAP **некоторых** из ASN.1-нотаций (X.519) текстовым представлением (LDAP URL и поисковые фильтры). В этом вопросе IETF не снискала нашей безграничной благодарности: значительное большинство ASN.1-нотаций всё ещё остаются в прежнем виде.

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

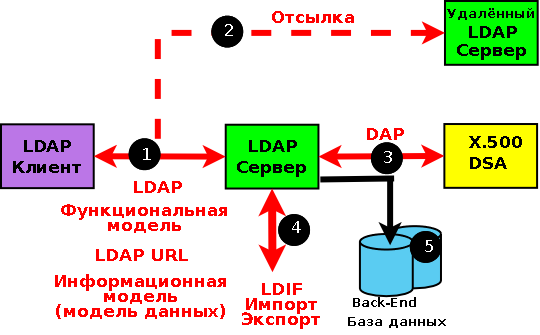
**2.2 Обзор LDAP**

Технически, LDAP — это всего лишь протокол, определяющий методы, посредством которых осуществляется доступ к данным каталога. Он также определяет и описывает, как данные **представлены** в службе каталогов (**Модель данных** или **Информационная модель**). Наконец, он определяет, каким образом данные загружаются (импортируются) и выгружаются (экспортируются) из службы каталогов (с использованием [LDIF](https://pro-ldap.ru/tr/zytrax/apd/index.html#ldif)). LDAP не определяет, **как** происходит хранение и манипулирование данными. С точки зрения стандарта хранилище данных и методы доступа к нему — это **"чёрный ящик"**, за который, как правило, отвечают модули **back-end** (механизмы манипуляции данными) какой-либо конкретной реализации LDAP (обычно в них используется некоторая форма транзакционной базы данных).

LDAP определяет четыре **модели**, которые мы перечислим и кратко обсудим, а затем Вы можете спокойно забыть о них, поскольку для понимания LDAP они мало что дают.

1. **Информационная модель:** мы склонны использовать термин **модель данных**, на наш взгляд он более интуитивный и понятный. Модель данных (или информационная модель) определяет, каким образом информация или данные **представлены** в системе LDAP. Это может совпадать или не совпадать с фактическим методом представления данных в хранилище на физическом носителе. Как упоминалось выше, вопрос хранилищ данных лежит за пределами стандартов LDAP.
2. **Модель именования:** определяет все вещи наподобие 'dc=example,dc=com', с которыми Вы сталкиваетесь в системах LDAP. Здесь мы максимально придерживаемся спецификации, поскольку эти термины используются очень широко.
3. **Функциональная модель:** при чтении, поиске, записи или модификации LDAP Вы используете функциональную модель — кто бы мог подумать!
4. **Модель безопасности:** Вы можете контролировать, причём весьма детально, кто, что и с какими именно данными может сделать. Это сложная, но мощная штука. Мы постепенно внедрялись в данную концепцию и посвятили ей отдельную [главу](https://pro-ldap.ru/tr/zytrax/ch15). На начальном этапе можно забыть о безопасности. Вы всегда можете вернуться и модернизировать безопасность в LDAP. Если модернизация впоследствии будет невозможна, мы будем описывать реализацию безопасности по тексту. Эта модель также включает в себя защиту данных при передаче по сети, такую как [TLS/SSL](https://pro-ldap.ru/tr/zytrax/tech/ssl.html). Хорошая, но на редкость сложная штука.

Сфера стандартов LDAP показана на диаграмме ниже. Обозначенные красным вещи (1, 2, 3, 4) определены в протоколе (различных RFC, определяющих LDAP). Происходящее же в "чёрных ящиках" (или, в данном случае, в зелёных, жёлтых и сиреневых ящиках), а также показанное чёрной стрелкой обращение к базам данных (5) выходит за рамки стандартов.



Каждый компонент кратко описан здесь, немного подробнее ниже и в мельчайших деталях в последующих главах. Но для начала четыре важных момента:

1. LDAP не определяет, каким образом данные хранятся, только каким образом к ним осуществляется доступ. НО большинство реализаций LDAP используют стандартные базы данных в качестве механизмов манипуляции данными, и лишь OpenLDAP предлагает выбор поддерживаемых механизмов манипуляции данными.
2. Когда Вы общаетесь с сервером LDAP, Вы понятия не имеете, откуда поступают данные: на самом деле одна из ключевых задач стандарта — скрывать такой уровень детализации. Теоретически, данные могут поступать из одной ИЛИ НЕСКОЛЬКИХ локальных баз данных, либо одной или нескольких служб X.500 (в наши дни это большая редкость). Откуда и каким образом Вы будете получать данные — это детали реализации, они важны только на этапе определения рабочей конфигурации Вашего LDAP-сервера (серверов).
3. Две концепции, — доступ к службе LDAP и эксплуатация службы LDAP, — должны быть чётко разделены в Вашем сознании. Когда Вы проектируете каталог, сконцентрируйтесь на том, чего Вы хотите от него добиться, для чего он будет предназначен (организация данных) и не думайте о реализации. Затем, во второй фазе, во время составления рабочей конфигурации LDAP, сконцентрируйтесь на том, где данные будут располагаться, какие будут применяться системы хранения.
4. Ряд коммерческих продуктов баз данных предоставляет LDAP-представление (LDAP-обёртку или LDAP-абстракцию) реляционной базы данных или базы данных другого типа.

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.3 LDAP и базы данных**

LDAP характеризуется как сервис "один раз записал — много раз прочитал". Другими словами, от данных, обычно хранящихся в каталоге LDAP, не ожидается, чтобы они менялись при каждом доступе. Для иллюстрации: LDAP не подходит для ведения учета банковских операций, так как они, по своей природе, изменяются почти при каждом доступе (бизнес-транзакции). С другой стороны, LDAP как нельзя лучше подходит для учёта различных аспектов деятельности банка, меняющихся значительно реже: списка отделений, часов работы, сотрудников и т.д.

**Оптимизация на чтение**

Во фразе "один раз записал — много раз прочитал" до конца не ясно, насколько много это **много**?

Где проходит грань разумного использования между LDAP и классическими, ориентированными на транзакции реляционными базами данных, к примеру, SQLite, MySQL, PostGreSQL? Если обновление происходит при каждом втором доступе, будет ли разумно использовать в таком приложении LDAP, или нужно, чтобы обновления происходили раз в тысячу или в миллион обращений?

Литература по этой теме крайне редка и имеет тенденцию придерживаться таких беспроигрышных LDAP-приложений, как адресные книги, которые изменяются, возможно, раз в столетие.

Простого ответа на этот вопрос нет, однако следующие замечания могут оказаться полезными:

1. Увеличение нагрузки при операциях записи происходит из-за обновления индексов. Чем больше индексов (для ускорения поиска), тем, по возможности, реже должны выполняться обновления каталога. Соотношение чтение:запись в хорошо оптимизированных на чтение каталогах должно быть 1000:1 и даже больше. Для умеренно оптимизированных каталогов (2-3 индекса) разумным будет соотношение 500:1 и выше.
2. Репликация LDAP генерирует несколько транзакций для каждого обновления, таким образом, желательно снизить практическую нагрузку, связанную с обновлениями (следует ориентироваться на соотношение чтение:запись 500:1 или выше).
3. Если объём данных велик (скажем, больше 100 000 записей), то даже при небольшом количестве индексов время обновления может быть значительным. Поэтому желательно снизить количество обновлений до минимально возможного (10 000:1 или выше).
4. Если объём данных относительно невелик (скажем, меньше 1000 записей, чего обычно хватает для большинства стандартных применений каталогов LDAP), индексов немного (не более 2-3) и отсутствует репликация, мы не видим рационального объяснения, почему Вы не можете использовать LDAP в форме, приближенной к транзакционной системе, то есть соотношение чтение:запись 5:1 или 10:1. При добавлении репликации более подходящим будет соотношение 50:1 или 100:1.
5. Мы полагаем, что настоящий ответ на этот вопрос (с уважением к памяти ушедшего от нас [Douglas Noel Adams](http://www.phnet.fi/public/mamaa1/adams.htm)): оптимальное соотношение количества чтений к количеству записей составляет 42!

**Представление организации данных**

Применяемые для доступа к каталогам LDAP примитивы (элементы протокола LDAP) используют модель данных, которая (возможно) абстрагирована от её физической организации. Эти примитивы обеспечивают **представление** данных в виде объектной модели и не заботятся об актуальной структуре данных. В действительности, относительная простота LDAP происходит именно от этой характеристики. Конкретная реализация сервера LDAP будет выполнять отображение примитивов LDAP в физическую организацию данных, используя свой **механизм манипуляции данными** как **"чёрный ящик"** в чистом виде.

В этом резкий контраст со, скажем, SQL, поскольку в SQL-запросах, используемых для получения данных, присутствует полное и детальное представление структур данных, их организация в таблицы, объединения, наличие первичных ключей и т.д.

**Синхронизация и репликация данных**

Реляционные и транзакционные базы данных идут на крайние меры для обеспечения целостности данных во время циклов записи/обновления, используя такие техники, как транзакции, блокировки, откаты и другие методы. Для такого типа технологий баз данных это жизненно важные и необходимые требования. Данная экстремальная форма синхронизации данных также действует при репликации данных на несколько хостов или серверов. Все представления данных постоянно будут соответствовать друг другу.

Главный LDAP-сервер и подчинённые ему серверы (или равноправные ему серверы в среде с несколькими главными серверами (multi-master)) используют простой асинхронный процесс репликации данных. В связи с этим во время цикла репликации возможна рассинхронизация данных в главной и подчинённой (или равноправной) системах. Во время этого (обычно очень короткого) периода времени запрос к главному и подчинённому серверам **может** дать разные результаты. Если вследствие такого расхождения мир с треском расколется пополам, значит для подобного приложения LDAP не подойдёт. Если же Bob Smith на несколько секунд (или даже меньше) будет числиться в бухгалтерии на одном LDAP-сервере, а на другом — в отделе продаж, вряд ли это кого-то сильно огорчит. В эту категорию попадает удивительно большое количество приложений.

**Примечание:** Современные реализации LDAP, особенно те, которые поддерживают конфигурации с несколькими главными серверами (Multi-master), становятся всё более изощренными в репликации обновлений. Кроме того, высокоскоростные сети связи позволяют значительно быстрее выполнять операции репликации. Однако, подобные решения всего лишь уменьшают промежуток времени, в течение которого две какие-либо системы будут рассинхронизированы, они не устраняют саму природную особенность LDAP, заключающуюся в возникновении рассинхронизации, даже если в большинстве современных реализаций таковая длится всего лишь доли секунды.

**2.3.1 Использование LDAP — резюме**

Так в чём же преимущества LDAP (каталогов), и почему каждый здравомыслящий человек будет их использовать?

Прежде чем попытаться ответить на этот вопрос, давайте абстрагируемся от тактических соображений производительности. В целом, реляционные СУБД всё ещё значительно быстрее реализаций LDAP. По мере разработки служб каталогов второго поколения это положение меняется, и, хотя реляционные СУБД всегда будут оставаться быстрее LDAP, разрыв значительно сократился вплоть до точки, в которой различия становятся уже практически несущественными, если, конечно, Вы сравниваете подобное с подобным (единичные сетевые транзакции, а не обновление высокоиндексированного атрибута при каждой операции — в этом случае, не обессудьте, Вы получите (или не получите) ровно столько, сколько заслужили).

Так почему же нужно использовать LDAP? Вот наш список ключевых характеристик, которые делают высокий (всё ещё) уровень боли терпимым:

1. LDAP предоставляет стандартизированные как удалённый, так и локальный методы доступа к данным. Таким образом, вполне реально заменить одну реализацию LDAP на другую, совершенно не влияя на внешний интерфейс доступа к данным. Реляционные СУБД в основном реализуют стандарты локального доступа, такие как SQL, но удалённые интерфейсы всегда остаются проприетарными.
2. Поскольку в LDAP используются стандартизированные методы доступа к данным, клиенты и серверы LDAP могут разрабатываться отдельно или быть получены из разных источников. В продолжении этой темы, LDAP может быть использован для абстрактного представления данных, содержащихся в ориентированных на транзакции базах данных, скажем, для выполнения пользовательских запросов, позволяя пользователям прозрачно (для LDAP-запросов) обращаться к разным поставщикам транзакционных баз данных.
3. LDAP предоставляет метод, посредством которого данные могут быть перемещены (делегированы) в несколько мест, не затрагивая при этом внешнего доступа к этим данным. Используя метод отсылок LDAP, данные могут быть перемещены на альтернативные LDAP-серверы путём только изменения операционных параметров. Таким образом, можно построить распределенные системы, возможно, с данными, поступающими из отдельных автономных организаций, обеспечивая при этом для своих пользователей единое, последовательное представление этих данных.
4. Системы LDAP могут быть настроены на репликацию данных на один или несколько серверов LDAP или одному или нескольким приложениям только за счет операционных параметров, без необходимости добавления дополнительного кода или изменения внешнего доступа к данным.

Эти характеристики концентрируют внимание исключительно на стандартной природе доступа к данным LDAP, не учитывая отношения количества чтений к количеству записей, которое, как отмечалось выше, зависит от количества обслуживаемых операционных индексов. Они неявно сбрасывают со счетов использование систем LDAP для обработки транзакций, хотя есть тенденция, что некоторые реализации LDAP учитывают такие возможности.

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.4 Информационная модель (модель данных или объектная модель) LDAP**

Каталоги LDAP используют модель данных, которая **представляет** данные как иерархию объектов. Это не означает, что LDAP является объектно-ориентированной базой данных. Как уже отмечалось выше, LDAP сам по себе является протоколом, позволяющим получить доступ к службам LDAP, и не определяющий, каким образом данные хранятся, — но операционные примитивы (чтение, удаление, модификация) работают с моделью (описанием/представлением) данных, имеющих (в основном) объектоподобные характеристики.

**2.4.1 Структура дерева объектов**

В этом разделе определяется сущность LDAP. Если Вы поймёте этот раздел и различные термины и отношения, с ним связанные, Вы поймёте LDAP.

В LDAP-системе данные представлены как иерархия объектов, каждый из которых называется [записью](https://pro-ldap.ru/tr/zytrax/apd/index.html#entry). Полученная в результате древовидная структура называется [Информационным деревом каталога (Directory Information Tree, DIT)](https://pro-ldap.ru/tr/zytrax/apd/index.html#dit). Верхнюю часть данного дерева обычно называют [**корнем (root)**](https://pro-ldap.ru/tr/zytrax/apd/index.html#root), (а также **базой (base)** или **суффиксом (suffix)**).

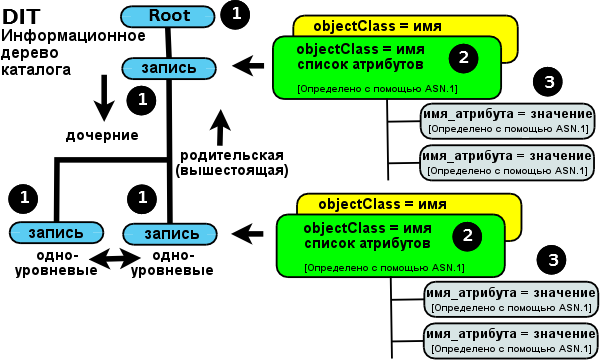
У каждой **записи** есть одна родительская запись (объект) и ноль или более дочерних записей (объектов). Каждая дочерняя запись (объект) является одноуровневой (братской) по отношению к другим дочерним записям своей родительской записи.

Каждая запись состоит из (является экземпляром) одного или нескольких [объектных классов (objectClass)](https://pro-ldap.ru/tr/zytrax/apd/index.html#objectclass). **Объектные классы** содержат ноль или более [атрибутов (attribute)](https://pro-ldap.ru/tr/zytrax/apd/index.html#attributes). Атрибуты имеют имена (и, иногда, аббревиатуры или псевдонимы) и обычно содержат данные (наконец-то!).

Характеристики (свойства) **объектных классов** и их **атрибутов** описываются определениями [ASN.1](https://pro-ldap.ru/tr/zytrax/apd/index.html#asn1).

Уф! Теперь Вы знаете всё, что только можно знать о LDAP. Остальное — детали. Да, этих деталей, пожалуй, очень много. Но суть LDAP именно в этом.

Представленная ниже диаграмма иллюстрирует эти отношения:



Информационная модель (модель данных) LDAP DIT

**Подытожим:**

1. Каждая **запись** (1) состоит из одного или нескольких **объектных классов** (2).
2. У каждого **объектного класса** (2) есть имя. Объектный класс представляет собой контейнер для атрибутов (в его определении идентифицируются атрибуты, которые он может или должен содержать).
3. У каждого **атрибута** (3) есть имя, он является членом одного или нескольких **объектных классов** (2) и содержит данные.
4. При наполнении DIT каждая **запись** будет уникально идентифицирована в иерархии (относительно своей родительской записи) данными, которые содержатся в этой записи (в атрибутах, которые содержатся в её объектном классе (классах)).

Теперь можно смело брать выходной на остаток дня и хорошенько отпраздновать!

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.4.2 Объектные классы**

Объектные классы являются, по существу, контейнерами **атрибутов**. Они описываются с помощью [определений ASN.1](https://pro-ldap.ru/tr/zytrax/ch3/#objectclasses). У каждого **объектного класса** есть уникальное имя. Существует огромное число предопределённых **объектных классов**, в каждом из которых полным-полно **атрибутов** для почти всех возможных применений каталогов LDAP. Однако, само собой разумеется, что среди всех этих предопределённых объектных классов нет того, который Вам просто необходим! У **объектных классов** есть ещё три характеристики:

1. **Объектный класс** определяет, **должен (MUST)** ли входящий в него атрибут присутствовать в записи (обязательный атрибут), или он **может (MAY)** присутствовать (необязательный атрибут).
2. Каждый **объектный класс** принадлежит к определённому типу: он может быть структурным (STRUCTURAL), вспомогательным (AUXILIARY) или абстрактным (ABSTRACT) (детально эти типы описаны в [следующей главе](https://pro-ldap.ru/tr/zytrax/ch3/index.html#objectclasses)). На этом этапе достаточно знать, что в **записи** должен быть один и только один структурный (STRUCTURAL) **объектный класс** и может быть ноль или более вспомогательных (AUXILIARY) объектных классов.
3. **Объектный класс** может быть частью иерархии, в этом случае он **наследует** все характеристики своего родительского **объектного класса (классов)** (включая все содержащиеся в них атрибуты).

**Объектные классы** являются контейнерами и управляют тем, какие атрибуты могут быть добавлены к каждой записи. В целом же они, как правило, остаются вне поля зрения, когда речь идет о доступе и опросе (поиске) по DIT. Здесь на первый план выходят атрибуты и записи.

[Вот неполный список наиболее часто используемых **объектных классов** и их **атрибутов**](https://pro-ldap.ru/tr/zytrax/ape/).

[Больше (значительно больше) об объектных классах](https://pro-ldap.ru/tr/zytrax/ch3/index.html#objectclasses), — но только если Вы хорошо разобрались с данным материалом (в любом случае, мы будем разбираться с этим в следующей главе), — в противном случае продолжайте читать здесь.

**Об уникальности:** Каждое используемое в LDAP имя является уникальным. Имя каждого объектного класса уникально среди объектных классов, но на этом дело не заканчивается. Уникальное имя объектного класса (или имя любого другого элемента LDAP) также является глобально уникальным именем во всём LDAP. Например, существует объектный класс с уникальным именем **person** (с ним мы будем пересекаться позже), но это имя также является глобально уникальным именем во всём LDAP. Не существует атрибута (или любого другого элемента) с именем **person**.

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.4.3 Атрибуты**

Каждый **атрибут** имеет имя (а также может иметь короткое имя или псевдоним) и обычно содержит данные. **Атрибуты** всегда связаны (являются членами) с одним или несколькими **объектными классами**. У **атрибутов** есть ряд интересных особенностей:

1. Все **атрибуты** являются членами одного или нескольких **объектных классов.**
2. Каждый **атрибут** определяет [тип данных](https://pro-ldap.ru/tr/zytrax/apa/types.html), которые он может содержать (ключевое слово SYNTAX в определении атрибута).
3. **Атрибуты** могут быть частью иерархии, в этом случае дочерний атрибут наследует все характеристики родительского атрибута. В данном случае иерархия атрибутов используется для упрощения и сокращения определений атрибутов (в ASN.1) там, где у нескольких атрибутов имеются одинаковые общие свойства, такие как максимальная длина, чувствительность/нечувствительность к регистру символов, или что-то ещё. Никакого другого смысла в иерархии атрибутов нет.
4. **Атрибуты** могут быть необязательными (ключевое слово MAY) или обязательными (ключевое слово MUST), согласно определениям ASN.1 того **объектного класса**, членами которого они являются. **Атрибут** может быть необязательным в одном **объектном классе** и обязательным в другом. Это свойство определяется в рамках **объектного класса**.

В разных местах этой документации в записях-примерах подобраны совершенно различные атрибуты, что может показаться путаницей. На самом же деле так происходит из-за необязательного характера большинства атрибутов. Это позволяет при составлении записей использовать подход "выбирай и соединяй": находим нужный нам **атрибут**, находим **объектный класс**, членом которого является этот атрибут (таких классов может быть несколько), и надеемся, что все остальные **атрибуты** из данного **объектного класса**, которые мы не хотим использовать, окажутся необязательными! Чтобы яснее это себе представить, попробуйте посмотреть [здесь](https://pro-ldap.ru/tr/zytrax/ape/index.html#attributes).

1. У **атрибутов** может быть одно (SINGLE-VALUE) или несколько (MULTI-VALUE) значений (как описано в их [определениях ASN.1](https://pro-ldap.ru/tr/zytrax/ch3/#attributes)). SINGLE-VALUE означает, что только одно значение данных может быть задано для этого **атрибута**. MULTI-VALUE означает, что этот **атрибут** может появляться в записи несколько раз с разными значениями данных. Если **атрибут** описывает, скажем, адрес электронной почты, может быть одно, два или 500 включений этого атрибута в запись, каждое с разным адресом электронной почты (атрибут многозначный (MULTI-VALUE)) — это один из ряда методов работы с почтовыми псевдонимами, применяемых при построении каталога. Значением по умолчанию для атрибута является MULTI-VALUE (что позволяет иметь несколько значений).
2. У **атрибутов** есть имя и, иногда, псевдоним (как описано в их определениях ASN.1), например, атрибут с именем **cn** является членом **объектного класса**, называемого **person** (и многих других), и имеет псевдоним **commonName**. Для ссылки на этот **атрибут** может использоваться как **commonName**, так и **cn**.
3. На всех уровнях иерархии данные, содержащиеся в каком-то из **атрибутов**, могут использоваться для однозначной идентификации **записи**. Это может быть любой **атрибут** в [записи](https://pro-ldap.ru/tr/zytrax/apd/index.html#entry). Это может быть даже комбинация двух или более атрибутов.

**Атрибут (атрибуты)**, выбранные для хранения данных, составляющих уникальность записи, иногда называются [атрибутом (атрибутами) именования](https://pro-ldap.ru/tr/zytrax/apd/index.html#naming-attribute) или [относительным уникальным именем (Relative Distinguished Name, RDN)](https://pro-ldap.ru/tr/zytrax/apd/index.html#rdn) — но подробнее об этом [в следующем разделе](https://pro-ldap.ru/tr/zytrax/ch2/#data).

[Взгляните на некоторые распространённые **объектные классы** и **атрибуты**](https://pro-ldap.ru/tr/zytrax/ape/). На данном этапе всё это выглядит очень страшно. Просто забудьте о том, что Вы только что видели, и продолжим чтение.

[Больше (значительно больше) об атрибутах](https://pro-ldap.ru/tr/zytrax/ch3/index.html#attributes), — но только если Вы хорошо разобрались с данным материалом (в любом случае, мы будем разбираться с этим в следующей главе), — в противном случае продолжайте читать здесь.

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.4.4 Описание дерева путём добавление записей (данных)**

В конце концов мы хотим поместить немного данных в наш каталог и использовать-таки эту замороченную штуку.

Описание древовидной структуры и первоначальное наполнение данными осуществляется путем добавления записей (с ассоциированными с ними объектными классами и атрибутами), начиная от корневой записи DIT и двигаясь вниз по иерархии. Таким образом, родительская запись всегда должна быть добавлена перед тем, как пытаться добавить дочерние записи.

Записи состоят из одного или нескольких объектных классов (единственного структурного объектного класса и нуля или более вспомогательных классов), которые выступают в качестве контейнеров для атрибутов. Именно атрибуты, а не объектные классы, содержат данные.

Ранее мы определили, что при создании/наполнении DIT каждая запись будет уникально идентифицироваться (относительно своей родительской записи) в иерархии. Единственным уникальным элементом любой структуры данных являются данные. Чтобы однозначно идентифицировать запись, нам нужно определить уникальные данные среди тех, которые содержатся в ней. Данные, содержащиеся в записи, определяются в атрибутах (присутствующих в объектных классах), таким образом, нам нужно определить атрибут, содержащий данные, являющиеся уникальными. Напомним, что многие атрибуты являются многозначными, — они могут присутствовать в записи несколько раз с разным содержимым, — поэтому для создания абсолютной, несомненной уникальности нам нужно определить сразу и атрибут, и содержащиеся в нём данные. Это делается с использованием формата **имя\_атрибута=значение\_(или\_данные)**, который в терминологии LDAP зовётся *утверждением значения атрибута* (Attribute Value Assertion, AVA).

Для иллюстрации, если в данной записи (на данном уровне иерархии) уникальными данными является слово **fred** (да-да, мы знаем, пример так себе, маловероятно, чтобы это слово было уникальным, но вдруг мы находимся в Узбекистане?), и оно содержится в атрибуте с именем [cn](https://pro-ldap.ru/tr/zytrax/ape/core-schema.html" \l "cn), то нашим AVA (утверждением значения атрибута) станет **cn=fred**. В данном случае, если мы хотим пооригинальничать или нам просто некуда девать время, мы можем записать его иначе: **commonName=fred** ([у атрибута cn есть псевдоним commonName](https://pro-ldap.ru/tr/zytrax/ape/core-schema.html#cn)).

Но вдруг так случится, что cn=fred не будет абсолютно уникальным (Кто там хихикает на заднем плане? Мы всё слышим!), а значит не сможет быть уникальным идентификатором для данной записи. Мы можем либо изменить выбранное нами значение (в записи может оказаться несколько значений cn=value, из которых мы можем выбрать), либо изменить выбранный нами атрибут (может неплохо подойти [sn=de Gamma](https://pro-ldap.ru/tr/zytrax/ape/core-schema.html" \l "sn), если, конечно, мы не в Португалии). Кроме того, мы можем использовать второе AVA для обеспечения уникальности. В этом случае мы сохраним наше **cn=fred**, но добавим AVA [drink=tamarind juice](https://pro-ldap.ru/tr/zytrax/ape/cosine.html" \l "favouritedrink) (внесём элемент экзотики в наши серые будни). Тогда уникальное значение будет записываться как **cn=fred+drink=tamarind juice**. Уникальней просто некуда.

Добавление записей может происходить разными путями, один из которых — использовать файлы формата **LDAP Data Interchange Files (LDIF)**, который полностью описан в [одной из последующих глав](https://pro-ldap.ru/tr/zytrax/ch8). Файлы LDIF — это текстовые файлы, описывающие древовидную иерархию — [информационное дерево каталога (Directory Information Tree, DIT)](https://pro-ldap.ru/tr/zytrax/apd/index.html#dit), и данные, добавляемые в каждый атрибут. Ниже приведён простой пример LDIF-файла, описывающий корневое DN (dc=example,dc=com) и добавляющий три дочернии записи в ветку **people**.

**Примечания:**

1. На данном этапе не так важно досконально разбираться во всех значениях этого LDIF-файла. В примерах [главы 5](https://pro-ldap.ru/tr/zytrax/ch5) охвачены подробности настройки LDIF-файлов, а в [главе 8](https://pro-ldap.ru/tr/zytrax/ch8) LDIF-файлы разъясняются в красочных деталях. На данном этапе достаточно знать, что LDIF-файлы могут использоваться для создания DIT и выглядят примерно так, как приведено ниже (данный LDIF создаёт DIT со структурой, приведённой на русинке 2.4.4-1).
2. Добавление LDAP-записей может быть также выполнено с помощью LDAP-клиентов, таких как [LDAP-браузеры](https://pro-ldap.ru/tr/zytrax/apb/#tools) общего назначения (смотрите [LDAPBrowser/Editor](https://pro-ldap.ru/tr/zytrax/ch14/ldapbrowser.html)) или специализированные приложения.

version: 1

## указание версии не является строго необходимым (и некоторые реализации его отклоняют),

## но, в общем случае, это хорошая практика

## Определяем DIT ROOT/BASE/SUFFIX ####

## используется формат RFC 2377 (доменные имена)

## dcObject - это ВСПОМОГАТЕЛЬНЫЙ объектный класс и, кроме него, запись

## ДОЛЖНА иметь СТРУКТУРНЫЙ объектный класс (в данном случае, organization)

# это последовательность ЗАПИСИ и ей предшествует ПУСТАЯ СТРОКА

dn: dc=example,dc=com

dc: example

description: Лучшая компания в целом свете

objectClass: dcObject

objectClass: organization

o: Example, Inc.

## ПЕРВЫЙ уровень иерархии - люди (people)

# это последовательность ЗАПИСИ, она должна предваряться ПУСТОЙ строкой

dn: ou=people, dc=example,dc=com

ou: people

description: Все люди в организации

objectClass: organizationalUnit

## ВТОРОЙ уровень иерархии - записи людей

# это последовательность ЗАПИСИ, она должна предваряться ПУСТОЙ строкой

dn: cn=Robert Smith,ou=people,dc=example,dc=com

objectclass: inetOrgPerson

cn: Robert Smith

cn: Robert

sn: Smith

uid: rsmith

mail: robert@example.com

mail: r.smith@example.com

ou: sales

## ВТОРОЙ уровень иерархии - записи людей

# это последовательность ЗАПИСИ, она должна предваряться ПУСТОЙ строкой

dn: cn=Bill Smith,ou=people,dc=example,dc=com

objectclass: inetOrgPerson

cn: Bill Smith

cn: William

sn: Smith

uid: bsmith

mail: bill@example.com

mail: b.smith@example.com

ou: support

## ВТОРОЙ уровень иерархии - записи людей

# это последовательность ЗАПИСИ, она должна предваряться ПУСТОЙ строкой

dn: cn=John Smith,ou=people,dc=example,dc=com

objectclass: inetOrgPerson

cn: John Smith

sn: smith

uid: jsmith

mail: jim@example.com

mail: j.smith@example.com

ou: accounting

**Важное замечание:** Строки в приведённом выше LDIF-файле, начинающиеся с 'dn:', по существу сообщают LDAP-серверу, каким образом структурировать или располагать запись в DIT (об этом в следующем разделе). В общем случае не важно, значение какого атрибута используется для этой цели, при условии уникальности 'dn:'. В первой записи ВТОРОГО уровня (третьей записи LDIF-файла) приведённого примера для этой цели было выбрано "dn: cn=Robert Smith, ou=people, dc=example, dc=com". Точно также могло быть выбрано, скажем, "dn: uid=rsmith, ou=people, dc=example, dc=com". При LDAP-поиске может применяться любая комбинация атрибутов и запись будет найдена независимо от того, какое значение было использовано в 'dn:' при её создании. Однако, если запись планируется использовать для аутентификации пользователя, скажем, для входа в систему или организации технологии единого входа (Single Sign-On), значение 'dn:' становится чрезвычайно важным и определяет идентификатор входа в систему (в качестве которого, как правило, используется uid), или, на жаргоне, DN подсоединения. Иногда (особенно в контексте LDAP, используемого в Microsoft AD) имя записи называется DN принципала (Principal DN), хотя этот термин не используется в определениях стандартов LDAP. [Дополнительная информация по этой теме](https://pro-ldap.ru/tr/zytrax/apa/binddn.html).

Если данное примечание кажется вам белибердой, можете пока забыть о нём. Позже мы это подробнейшим образом осветим.

Мы расскажем о [LDIF](https://pro-ldap.ru/tr/zytrax/ch8)-файлах позже, поскольку они нам ещё понадобятся, а приведённый выше LDIF определяет следующую структуру:

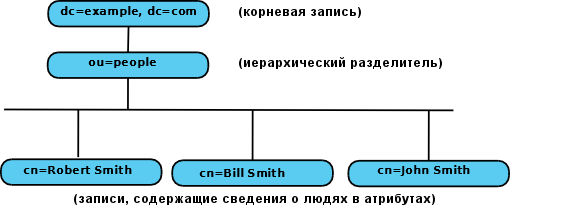


Рисунок 2.4.4-1: Структура DIT, созданная LDIF-файлом

После того, как DIT определён и запущен в работу, в дальнейшем информацию в него можно добавлять с помощью LDIF, LDAP-браузера, web-интерфейса или другого программного интерфейса.

Используя LDIF-файлы, данные также можно экспортировать (сохранить) в качестве резервной копии или для других целей.

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.4.5 Навигация по дереву**

После того, как мы поместили данные в наше дерево (DIT), обычно самое время начать с ними работать!

Для этого мы должны посылать команды (на чтение, поиск, модификацию и т.д.) нашему LDAP-серверу, а чтобы сделать это, мы должны быть в состоянии сообщить LDAP-серверу, где находятся данные (для записи) или хотя бы где они примерно находятся (для поиска/чтения).

Короче говоря, мы должны осуществлять навигацию (ориентироваться) в каталоге.

Но для начала давайте введём ещё немного существенной терминологии.

В предыдущем разделе мы определили, что каждая запись должна быть уникально идентифицируемой (относительно своей родительской записи) с использованием одного (или нескольких) AVA (утверждений значения атрибута), например, **cn=fred** или, как в примере выше, **cn=Robert Smith**. Из правила, что каждое идентифицирующее AVA (или несколько AVA) должны быть уникальными (относительно родительской записи) в иерархии, следует, что путь к любой записи на любом уровне также должен быть уникальным (он представляет собой сумму индивидуально уникальных записей).

Терминология LDAP в области адресации заставляет задуматься об ограниченности английского языка. До сих пор для описания AVA записи мы использовали термин **уникальное (unique)**, достаточно распространённое и понятное английское слово. Мы можем также сказать, что каждая запись имеет **различное (different)** имя. Кроме того (начинается!), мы можем также несколько извилисто выразиться, что, в силу своей уникальной идентификации, каждая запись **отлична (distinguished)** от своих соседей. Мы у цели! Основоположники служб каталогов в своей бесконечной мудрости решили использовать слово "Distinguished" ("Отличительное", *обычно в русскоязычной документации переводится как "Уникальное" — примечание переводчика*).

Итак, AVA, например, cn=Robert Smith, уникально идентифицирующее запись, называется *относительным уникальным именем* (Relative Distinguished Name, RDN), то есть именем, уникальным *по отношению* к родительской записи. Путь от корневой записи DIT (она же базовая запись или суффикс) к данной записи представляет собой сумму всех RDN (соединённых через запятую (,) в порядке слева направо от младшей до старшей) называется *уникальным именем* (Distinguished Name, DN). Красота! Какими бы ни были Ваши взгляды на достоинства (или недостатки) LDAP и X.500, способность групп их стандартизации генерировать уникальную (или отличительную) терминологию не подлежит сомнению.

Для иллюстрации, в DIT из нашего примера путь от корневой записи до записи, уникально идентифицируемой AVA cn=Robert Smith, идёт от (RDN) dc=example,dc=com через (RDN) ou=people, и заканчивается на (RDN) cn=Robert Smith. Итоговое DN будет записываться как **cn=Robert Smith,ou=people,dc=example,dc=com**.

Три дополнительных момента, пока Вы ещё способны воспринимать информацию.

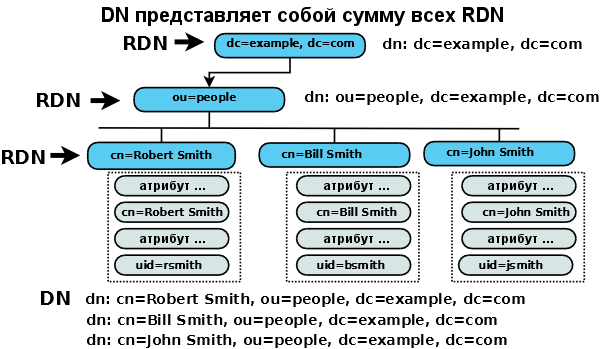
Напоминаем, что для создания уникального идентификатора можно применять несколько AVA, в этом случае RDN может выглядеть так: **cn=Robert Smith + uid=rsmith** (обычно это называется многозначным RDN). Эквивалентное ему DN будет выглядеть так: **cn=Robert Smith + uid=rsmith, ou=peope,dc=example, dc=com** (наличие или отсутствие пробелов между RDN не существенно).

DN записи ou=people: **ou=people,dc=example,dc=com**. DN описывает путь к любой записи в DIT.

В **dc=example, dc=com**, очевидно, два RDN (dc=example и dc=com). Это стандартная и легитимная конструкция ([в дальнейшем мы её обсудим](https://pro-ldap.ru/tr/zytrax/apa/ldap-root.html)), используемая для определения корневой записи (она же базовая запись или суффикс). (Если Вы в настроении помудрствовать лукаво на эту тему, можно было бы назвать эту конструкцию много-RDN-ной RDN, но, честное слово, не стоит).

Для навигации по DIT мы можем задать путь (DN) к месту, где находятся наши данные (cn=Robert Smith, ou=people,dc=example, dc=com приведёт нас к уникальной записи), либо мы можем задать путь (DN) к месту, где, как мы предполагаем, находятся наши данные (скажем, ou=people,dc=example,dc=com), а затем выполнить поиск по паре или нескольким парам атрибут=значение, чтобы найти целевую запись (или записи). Если мы хотим выполнить изменения в каталоге (модификацию на жаргоне LDAP), как правило, следует указывать уникальную запись. Однако, если мы устраиваем DIT допрос с пристрастием, достаточно лишь примерной точности — мы получим всё, что соответствует [критериям поиска](https://pro-ldap.ru/tr/zytrax/apa/search.html).

Следующая диаграмма иллюстрирует DN и RDN:



[Дальнейшее объяснение с некоторыми рабочими примерами](https://pro-ldap.ru/tr/zytrax/apa/dn-rdn.html).

[Наверх](https://pro-ldap.ru/tr/zytrax/ch2/#contents)

**2.5 Отсылки и репликация LDAP**

Одним из наиболее мощных аспектов LDAP (и X.500) является вложенная в них способность делегировать ответственность за поддержание части каталога другому серверу, сохраняя при этом общую картину каталога как единого целого. Таким образом, можно создать в каталоге компании делегирование ответственности (**отсылку (referral)** в терминологии LDAP) той части всего каталога, в которой описан конкретный отдел, LDAP-серверу этого отдела. В этом аспекте LDAP почти полностью отражает концепцию делегирования DNS, если эта концепция Вам знакома.

В отличие от системы DNS, в стандартах не предусмотрена возможность сообщить LDAP-серверу проследовать по отсылке (разрешить отсылку), LDAP-клиенту оставлена возможность напрямую обращаться к новому серверу, используя возвращённую отсылку. Равным образом, поскольку в стандарте не определена организация данных LDAP, возможность перехода LDAP-сервера по ссылке (разрешение ссылки) не противоречит стандартам и некоторые LDAP-серверы выполняют эту функцию автоматически, используя процесс, обычно называемый **сцеплением (chaining)**.

OpenLDAP буквально следует стандартам и по умолчанию не выполняет **сцепление**, а всегда возвращает отсылку. Однако OpenLDAP может быть настроен на выполнение **сцепления** с помощью директивы [overlay chain](https://pro-ldap.ru/tr/zytrax/ch6/index.html" \l "overlay).

Встроенная функция репликации LDAP позволяет создать одну или несколько подчинённых копий каталога (DIT) от одной главной (и даже, в некоторых реализациях, распространять изменения между несколькими главными копиями DIT), таким образом, по сути, создаётся устойчивая структура.

Важно, однако, подчеркнуть разницу между LDAP и транзакционными базами данных. При выполнении обновления в главном каталоге LDAP, для обновления подчинённых копий каталога (или других главных копий в конфигурации с несколькими главными серверами) может потребоваться некоторое время (в компьютерной терминологии) — главный и подчинённый каталоги (или разные главные каталоги в конфигурации с несколькими главными серверами) могут быть рассинхронизированы в течение какого-то периода времени.

В контексте LDAP, временное отсутствие синхронизации DIT считается несущественным. В случае же транзакционной базы данных, даже временное отсутствие синхронизации считается катастрофическим. Это подчёркивает различия в характеристиках данных, которые должны храниться в каталоге LDAP и в транзакционной базе данных.

Конфигурация репликации и отсылок [рассматривается далее](https://pro-ldap.ru/tr/zytrax/ch7), а также в [приводимых примерах](https://pro-ldap.ru/tr/zytrax/ch5).

**2.5.1 Отсылки LDAP**

Рисунок 2.5-1 демонстрирует поисковый запрос с базовым DN dn:cn=thingie,o=widgets,dc=example,dc=com к LDAP-системе с отсылками, который полностью удовлетворяется первым LDAP-сервером (LDAP1):

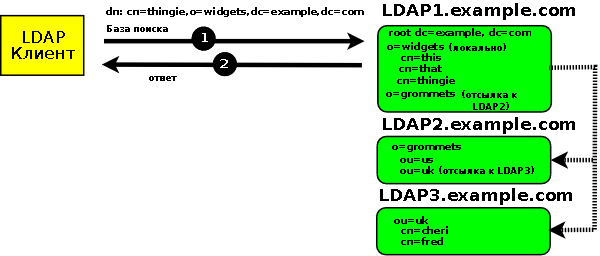


Рисунок 2.5-1 — Запрос, удовлетворяемый только от LDAP1

Рисунок 2.5-2 демонстрирует поисковый запрос с базовым DN dn:cn=cheri,ou=uk,o=grommets,dc=example,dc=com к LDAP-системе с отсылками, ответ на который возвращается в результате серии отсылок к серверам LDAP2 и LDAP3, а LDAP-клиенты всегда следуют по отсылкам:

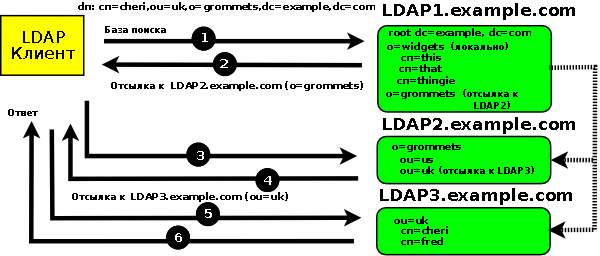


Рисунок 2.5-2 — Запрос, генерирующий отсылки к LDAP2 и LDAP3

**Примечания:**

1. Все клиентские запросы начинаются с обращения к глобальному каталогу LDAP1.
2. На сервере LDAP1 запросы любых данных, содержащие **widgets** в качестве [RDN](https://pro-ldap.ru/tr/zytrax/apd/index.html#rdn) в [DN](https://pro-ldap.ru/tr/zytrax/apd/index.html#dn), удовлетворяются непосредственно из LDAP1, например:
3. dn: cn=thingie,o=widgets,dc=example,dc=com
4. На сервере LDAP1 запросы любых данных, содержащие **grommets** в качестве [RDN](https://pro-ldap.ru/tr/zytrax/apd/index.html#rdn) в [DN](https://pro-ldap.ru/tr/zytrax/apd/index.html#dn), генерируют отсылку на сервер LDAP2, например:
5. dn: cn=cheri,ou=uk,o=grommets,dc=example,dc=com
6. На сервере LDAP2 запросы любых данных, содержащие **uk** в качестве [RDN](https://pro-ldap.ru/tr/zytrax/apd/index.html#rdn) в [DN](https://pro-ldap.ru/tr/zytrax/apd/index.html#dn), генерируют отсылку на сервер LDAP3, например:
7. dn: cn=cheri,ou=uk,o=grommets,dc=example,dc=com
8. Если LDAP-сервер сконфигурирован на выполнение сцепления (то есть на следование отсылкам, как показано альтернативными пунктирными линиями), то LDAP-клиенту будет отправлен один единственный ответ. Сцепление контролируется конфигурацией LDAP-сервера и значениями в поисковых запросах. Информация по сцеплению [здесь](https://pro-ldap.ru/tr/zytrax/ch7/referrals.html#chaining).
9. На рисунках показано явное сцепление с использованием объектного класса [referral](https://pro-ldap.ru/tr/zytrax/ape/core-schema.html" \l "referral). Серверы OpenLDAP могут быть [настроены так](https://pro-ldap.ru/tr/zytrax/ch6/#referral), чтобы возвращать общую отсылку в случае, если запрашиваемый DN не был найден во время операции поиска.

**2.5.2 Репликация LDAP**

Функции репликации позволяют копировать обновления LDAP DIT на одну или несколько LDAP-систем в целях резервирования и/или повышения производительности. В этом контексте стоит подчеркнуть, что репликация работает на уровне DIT, а не на уровне LDAP-сервера, поскольку на одном LDAP-сервере может обслуживаться несколько DIT. Репликация происходит периодически, в течение промежутка времени, известного как **время цикла репликации** (по сути, это время, необходимое для отправки обновленных данных на сервер-реплику и получения подтверждения об успешном завершении операции). В общем случае, существуют методы уменьшения времени цикла репликации с помощью настроек, но обычно они приводят к снижению производительности или увеличению нагрузки на сеть. Исторически для выполнения репликации в OpenLDAP использовался отдельный демон (slurpd), но, начиная с версии 2.3, стратегия репликации коренным образом изменилась, были достигнуты серьёзные улучшения в гибкости и возможностях настройки времени цикла репликации. Существует два возможных типа конфигураций репликации, у каждого из которых есть несколько вариантов.

1. **Главный-подчиненный (Master-Slave):** В конфигурации главный-подчинённый обновляется одно единственное DIT (на жаргоне OpenLDAP оно называется главным (master) или поставщиком репликации (provider)), и эти обновления реплицируются или копируются на один или несколько указанных LDAP-серверов, на которых запущены подчинённые DIT (на жаргоне OpenLDAP они называются потребителями репликации (consumer)). Подчинённые серверы оперируют с доступной только для чтения копией главного DIT. Пользователи, которые выполняют только чтение, будут превосходно себя чувствовать, работая с серверами, содержащими подчинённые DIT, а пользователи, которым нужно вносить изменения в каталог, должны обращаться к серверу, содержащему главное DIT. При определённых условиях конфигурация главный-подчинённый позволяет существенно сбалансировать нагрузку. Однако, у неё есть два очевидных недостатка:
   * Если всем (большинству пользователей) дана возможность (требуется) вносить изменения в DIT, то либо им потребуется получать доступ к одному серверу (с подчинённым DIT) для осуществления чтения и к другому серверу (с главным DIT) для выполнения обновлений, либо они всегда будут обращаться к серверу, на котором запущено главное DIT. В последнем случае репликация выполняет только функцию резервного копирования.
   * Поскольку существует только один сервер, содержащий главное DIT, он представляет собой единую точку отказа для операций записи, которая может повлиять на работоспособность всей системы (хотя, в случае серьёзного сбоя, подчинённое DIT может быть переконфигурировано для работы в качестве главного).
2. **Несколько главных (Multi-Master):** В конфигурации с несколькими главными серверами могут обновляться несколько главных DIT, запущенных на одном или нескольких серверах, и результаты обновлений распространяются на другие главные серверы.

Исторически OpenLDAP довольно долго не поддерживал операции с несколькими главными серверами, но в версии 2.4 окончательно введены такие возможности. В этом контексте, наверное, стоит отметить две вариации общей проблемы **конкуренции обновлений**, специфичной для всех конфигураций с несколькими главными серверами. Эта проблема выявлена проектом OpenLDAP для своей конфигурации с несколькими главными серверами, но она относится ко всем системам LDAP:

* + **Конкуренция значений**. Если выполняется два обновления одного и того же атрибута в одно и то же время (в пределах времени цикла репликации) и при этом атрибуту назначаются разные значения, то, в зависимости от типа атрибута (SINGLE или MULTI-VALUED), результирующая запись может быть в неправильном или непригодном для использования состоянии.
  + **Конкуренция удаления**. Если один пользователь добавляет дочернюю запись в то же самое время (в пределах времени цикла репликации), когда другой пользователь удаляет оригинальную запись, то удалённая запись вновь появится.

Рисунок 2.5-3 показывает несколько возможных конфигураций репликации и объединяет их с отсылками из предыдущего раздела, чтобы показать всю мощь и гибкость LDAP. Следует отметить, что большинство конфигураций LDAP не настолько сложны.

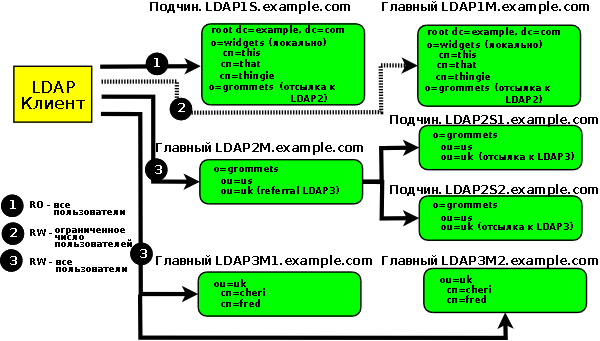


Рисунок 2.5-3 — Конфигурации репликации

**Примечания:**

1. **RO** = только чтение, **RW** = чтение-запись
2. В случае LDAP1 клиент взаимодействует с подчинённой системой, доступной только для чтения. Для выполнения операций модификации (записи) клиенты должны обращаться к главной системе.
3. В случае LDAP2 клиент взаимодействует с главной системой, которая реплицируется на две подчинённые.
4. LDAP3 является системой с несколькими главными серверами и для выполнения операций чтения (поиска) и/или записи (модификации) клиенты могут обращаться к любому серверу. В данной конфигурации каждый главный сервер может, в свою очередь, иметь одно или несколько подчинённых DIT.