فصل اول

معرفی سرویس دایرکتوری OpenLDAP

۱-۱ سرویس دایرکتوری چیست؟

دایرکتوری یک پایگاه داده اختصاصی است که برای خواندن، مرور و جست‌وجو بهینه‌سازی شده است.

نکته: یک دایرکتوری تعریف شده توسط برخی ابزارها، تنها یک پایگاه داده بهینه شده برای خواندن می‌باشد. این تعریف، در بهترین حالت، بسیار ساده است.

دایرکتوری‌ها معمولاً دارای اطلاعات توصیفی و ویژگی محور هستند و از قابلیت‌های فیلترسازی پیشرفته پشتیبانی می‌کنند. دایرکتوری‌ها عموماً از انتقال‌های پیچیده و یا طرح‌های roll-back موجود در سیستم‌های مدیریت پایگاه داده طراحی شده برای به‌روزرسانی‌های پیچیده با حجم بالا پشتیبانی نمی‌کنند. به‌روزرسانی‌های دایرکتوری‌ها معمولاً ساده هستند، یا همه‌چیز تغییر می‌کند و یا هیچ‌چیز، البته در صورتی که اصولاً اجازه تغییر داشته باشند. طراحی دایرکتوری‌ها طوری است که در عملیات جست‌وجو و ارجاع حجم بالا، پاسخگویی سریعی داشته باشند. دایرکتوری‌ها ممکن است برای افزایش دسترسی و قابلیت اطمینان، قابلیت تکرار اطلاعات را داشته باشند تا زمان پاسخ را کاهش

دهند. هنگامی که اطلاعات دایرکتوری replicate می‌شود، عدم ثبات موقت میان پاسخ‌ها ممکن است هیچ مشکلی به وجود نیاورد، البته در صورتی که نهایتاً همگام‌سازی[۱] شوند.

برای ارائه سرویس دایرکتوری، روش‌های بسیاری وجود دارد. شیوه‌های مختلف اجازه می‌دهند که اطلاعات متنوع در دایرکتوری ذخیره شوند، ملازمات مختلف برای چگونگیِ ارجاع به اطلاعات، پرس‌وجو و به‌روزرسانی، چگونگیِ محافظت از دسترسی غیرمجاز و ... را تعیین کنند. برخی از سرویس‌های دایرکتوری محلی هستند و به محیط محدودی خدمات ارائه می‌دهند (برای مثال خدمات اثر انگشت در یک دستگاه). دیگر سرویس‌ها جهانی هستند و به محیط بسیار بزرگی ارائه خدمات می‌کنند (برای مثال کل اینترنت). سرویس‌های جهانی معمولاً توزیع‌شده هستند، یعنی داده‌های آن‌ها بین چند دستگاه پخش شده است که همه آن‌ها برای ارائه خدمات دایرکتوری همکاری می‌کنند. سرویس‌های جهانی معمولاً یک فضای نام یکسان انتخاب می‌کنند که باعث می‌شود داده‌ها دارای ظاهر یکسانی باشند و مهم نیست که شما در رابطه با داده‌ها کجا قرار گرفته باشید.

وقتی اطلاعات دایرکتوری Replicate می‌شوند، عدم ثبات موقت بین Replicateها مشکلی ایجاد نمی‌کند، البته تا زمانی که عدم ثبات به‌موقع حل شود.

---

[۱] درعلم کامپیوتر، همزمان‌سازی یا هماهنگ سازی به دو مفهوم متمایز اما مرتبط هماهنگ سازی فرایندها و هماهنگ سازی داده‌ها اشاره می‌کند.
هماهنگ‌سازی فرایندها به این ایده که فرایندهای متعدد در یک نقطه خاص به هم می‌پیوندند و باید یک توافق یا تعهد برای انجام توالی خاصی از رفتار داشته باشند. معادل انگلیسی آن (synchronization) است.

یک دایرکتوریِ وب، مانند دایرکتوریِ ارائه شده توسط پروژه دایرکتوریِ باز <http://dmoz.org>، مثال خوبی از یک سرویس دایرکتوری است. این سرویس‌ها صفحات وب را فهرست‌بندی کرده و طراحیِ آن‌ها برای پشتیبانی از مرور و جست‌وجو می‌باشد.

با اینکه برخی سیستم نام دامنه اینترنت (DNS) را مثالی از سرویس دایرکتوریِ توزیع شده جهانی می‌دانند، اما DNS قابل مرور و جست‌وجو نیست. تعریف بهتری از آن، سرویس مراجعه[1] توزیع شده جهانی می‌باشد.

۱-۲ LDAP چیست؟

LDAP مخفف پروتکل دسترسی دایرکتوری سبک وزن (Lightweight Directory Access Protocol) است. همان‌طور که از نامش پیداست، LDAP یک پروتکل سبک‌وزن برای دسترسی به خدمات دایرکتوری است، به‌ویژه با سرویس‌های دایرکتوری مبتنی بر X.500. LDAP با TCP/IP یا دیگر سرویس‌های انتقالِ اتصال-محور کار می‌کند. جزئیات LDAP در RFC2251 «پروتکل دسترسی دایرکتوری سبک‌وزن (نسخه ۳)» و دیگر مستندات حاوی ویژگی‌های فنی RFC3377 تعریف شده است.

در این قسمت LDAP[2] از دیدگاه کاربر به‌صورت اجمالی بررسی می‌شود.

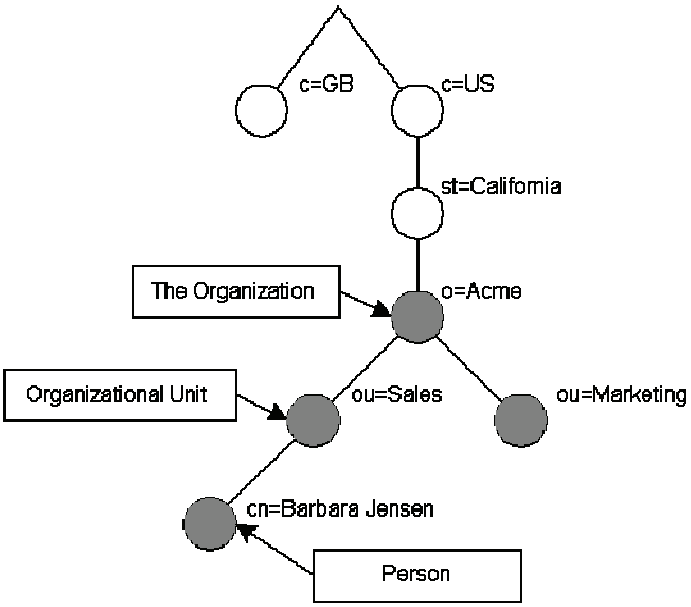
---

[1] Lookup

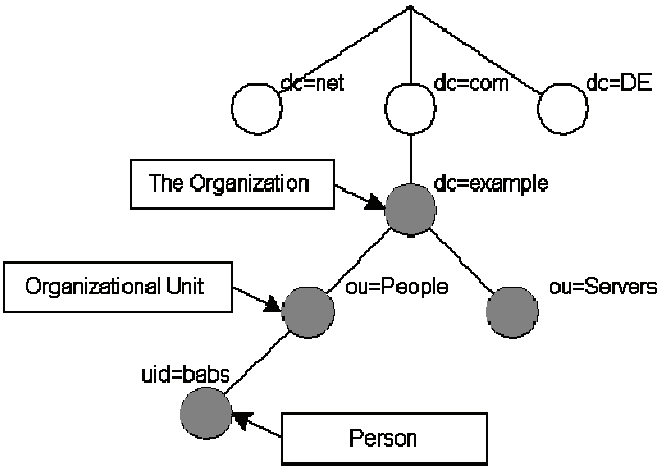
[2] اِل‌دَپ یا پروتکل دسترسی سبک وزن راهنما (به انگلیسی: Lightweight Directory Access Protocol (LDAP)) پروتکلی در شبکه‌های رایانه‌ای و درلایه کاربرد است که برای ارتباط با سرویس دایرکتوری استفاده می‌شود. این پروتکل در RFC 2251 و RFC 3377 مستندسازی شده است.

چه نوع اطلاعاتی را می‌توان در دایرکتوری ذخیره کرد؟ مدلِ اطلاعاتیِ LDAP بر پایه ورودی است. یک ورودی، مجموعه‌ای از ویژگی‌هاست که دارای نام متمایز (DN) منحصربه‌فرد جهانی است. به‌صورت انحصاری، از DN برای ارجاع به ورودی استفاده می‌شود. هرکدام از ویژگی‌های یک ورودی دارای «نوع» و یک یا چند مقدار است. نوع‌ها معمولاً رشته‌های حافظه هستند، مانند "CN" برای نام مشترک و یا "mail" برای آدرس ایمیل. ساختار مقدارها، به نوعِ ویژگی بستگی دارد. برای مثال یک ویژگیِ cn ممکن است دارای مقدار " Babs Jensen" باشد. ویژگی ایمیل می‌تواند مقدار babs@example.com را داشته باشد. ویژگیِ JpegPhoto ممکن است دارای یک عکس در فرمت Jpeg (باینری) باشد.

اطلاعات چگونه سازمان‌دهی شده‌اند؟ در LDAP، ورودی‌های دایرکتوری در ساختاری سلسله‌مراتبی درختی سازمان یافته‌اند. معمولاً، این ساختار نشان دهنده مرزهای سازمانی و یا جغرافیایی است. ورودی‌هایی که کشورها را نشان می‌دهند در بالای درخت قرار می‌گیرند و در زیر آن‌ها ورودی‌هایی قرار دارند که ایالت‌ها و سازمان‌های ملی را نشان می‌دهند. در زیر آن‌ها، ورودی‌ها ممکن است نشانگر واحدهای سازمان، افراد، پرینترها، اسناد و یا هر چیز دیگری باشند. شکل زیر مثالی از دایرکتوری درختی LDAP را با استفاده از نام‌گذاری سنتی نشان می‌دهد.



این درخت همچنین می‌تواند بر اساس نام‌های دامنه اینترنتی سازمان یابد. با توجه به اینکه این رویکرد برای نام‌گذاری، این امکان را فراهم می‌کند که سرویس دایرکتوری با استفاده از DNS مکان‌یابی شود، بنابراین محبوبیت بیشتری را از آن خود کرده است. شکل زیر مثالی از درخت دایرکتوری LDAP را با استفاده از رویکرد نام‌گذاریِ دامنه نشان می‌دهد.



به علاوه، LDAP به شما اجازه می‌دهد که ویژگی‌های مورد نیاز و مجاز در ورودی را با استفاده از یک ویژگیِ خاص به نام ObjectClass، کنترل کنید. مقدار ویژگیِ ObjectClass الگوی قوانینی را که آن ورودی باید از آن پیروی کند، مشخص می‌کند.

چگونه به اطلاعات ارجاع می‌شود؟ به یک ورودی، با استفاده از نام منحصر به فردش رجوع می‌شود که از نام خود ورودی (مشهور به نام متمایز نسبی (RDN)) و الحاق آن با نام ورودی‌های بالایی خود ساخته می‌شود. برای مثال، در نمونه‌ی نام‌گذاری اینترنتیِ بالا، نام RDN برای ورودی Jensenuid=babs و نام DN آن Narnara uid=babs,ou=People,dc=example,dc=com است. فرمت کامل DN در RFC2253، پروتکل دسترسی دایرکتوری سبک وزن (نسخه ۳): نمایش رشته UTF-8[1] نام‌های متمایز شرح داده شده است.

---

[1] یوتی‌اف-۸ (به انگلیسی: UTF-8) نوعی نویسه کدگذاری برای نوشتار است که فرمت ۸ بیت را رمزگذاری می‌کند و در مجموعه یونیکدهای اسکی طراحی شده و برای جلوگیری از مشکلات

دسترسی به اطلاعات چگونه است؟ LDAP برای پرس‌وجو و به‌روزرسانی دایرکتوری دارای عملگرهایی است. عملگرها برای اضافه یا حذف یک ورودی، تغییر ورودی‌های موجود و تغییر نام یک ورودی در دایرکتوری تعبیه شده‌اند. البته اکثر اوقات از LDAP برای جست‌وجوی اطلاعات در دایرکتوری استفاده می‌شود. عملیات جست‌وجوی LDAP این امکان را فراهم می‌کند که جست‌وجو برای ورودی‌ها در برخی از بخش‌های دایرکتوری مطابق با معیارهای مشخص شده توسط فیلتر جست‌وجو انجام شود. اطلاعات را می‌توان از هر ورودی که با معیار مشخص شده مطابقت دارد، فراخوانی کرد. برای مثال، شاید بخواهید در کل شاخه دایرکتوری و شاخه‌های زیرین dc=example, de=com افرادی که دارای نام Barbara Jensen هستند را جست‌وجو کرده و آدرس ایمیل هر ورودیِ یافته شده را دریافت کنید. با LDAP می‌توانید به راحتی این کار را انجام دهید. یا اینکه شاید بخواهید میان ورودی‌هایی که مستقیماً در زیر ورودی st=California, C=US قرار رفته‌اند به جست‌وجوی سازمان‌هایی بپردازید که عبارت Aceme در آن‌ها وجود داشته و دارای یک شماره فکس هستند. با LDAP می‌توانید این کار را انجام دهید. بخش بعدی جزئیات بیشتر کارهایی که می‌توانید با LDAP انجام داده و اینکه چطور می‌تواند برایتان مفید باشد را شرح می‌دهد.

اطلاعات چگونه از دسترسی غیر مجاز محافظت می‌شوند؟ برخی از سرویس‌های دایرکتوری هیچ‌گونه محافظتی ارائه نمی‌دهند و اجازه می‌دهند هرکسی اطلاعات را ببیند. LDAP دارای مکانیسمی برای احراز هویت کاربر است، به عبارت دیگر هویت او را برای سرور دایرکتوری

---

endianness در یوتی‌اف-۱۶ و یوتی‌اف-۳۲ ساخته شده است. بیش از نیمی از وبسایت‌ها در سراسر جهان از این یونی‌کد کدگذاری می‌شوند.

اثبات می‌کند و راه را برای کنترل دسترسی پیشرفته هموار می‌کند تا از اطلاعات موجود در سرور محافظت کند. LDAP همچنین از سرویس‌های محافظت از داده (ادغامی و محرمانه) پشتیبانی می‌کند.

۱-۳ چه وقت باید از LDAP استفاده کنیم؟

این سؤال بسیار خوبی است. در کل، شما باید وقتی از سرویس دایرکتوری استفاده کنید که می‌خواهید داده‌ها به‌صورت مرکزی مدیریت و ذخیره شوند و از طریق متدهای استاندارد قابل دستیابی باشند.

برخی از مثال‌های رایج در این کار شامل، نه محدود به این موارد می‌شود:

- احراز هویت ماشین
- احراز هویت کاربر
- گروه‌های کاربر/سیستم
- کتاب آدرس
- نمایش سازمانی
- مدیریت دارایی
- فهرست اطلاعات تلفن
- مدیریت منابع کاربر
- جست‌وجوی آدرس ایمیل
- فهرست پیکربندیِ برنامه
- فهرست پیکربندیِ PBX
- و ...

فایل‌های schema توزیع شده مختلفی وجود دارند که بر پایه استاندارد هستند، اما شما همیشه می‌توانید مشخصات schema خود را نیز ایجاد کنید.

همیشه روش‌های جدیدی برای استفاده از دایرکتوری و اعمال قوانین LDAP برای حل برخی مسائل وجود دارد، بنابراین هیچ پاسخ ساده‌ای برای این سؤال وجود ندارد.

اگر شک دارید، عضو انجمن عمومیِ LDAP برای بحث‌های غیرتجاری و مرتبط با اطلاعات LDAP در آدرس http://www.umich.edu/~dirsvcs/ldap/mailinglist.html شوید و سؤال خود را بپرسید.

۱-۴ چه وقت نباید از LDAP استفاده کنیم؟

وقتی که احساس می‌کنید باید دایرکتوری را تغییر دهید تا کار مورد درخواستتان را انجام دهد و یا اگر می‌خواهید یک برنامه از داده‌های شما استفاده کرده و آن را به کار بگیرد، احتمالاً طراحیِ دوباره‌ای مورد نیاز است (برای بحث بیشتر در مورد LDAP در برابر RDBMS، لطفاً بخش LDAP در برابر RDBMS را مطالعه کنید).

وقتی که LDAP ابزار مناسبی برای انجام یک کار باشد، کاملاً مشخص خواهد بود.

۱-۵ LDAP چگونه کار می‌کند؟

سرویس دایرکتوری LDAP بر پایه‌ی مدل کلاینت-سرور است. یک یا چند سرور LDAP حاویِ داده، درخت اطلاعاتی دایرکتوری را تشکیل می‌دهند. کاربر به سرورها متصل شده و یک سؤال از آن می‌پرسد. سرور با یک پاسخ و یا یک اشاره‌گر به سمت جایی که کاربر می‌تواند برای اطلاعات بیشتر مراجعه کند (معمولاً یک سرور LDAP دیگر) پاسخ می‌دهد. مهم نیست کاربر به کدام سرور LDAP متصل شود، سرور نمای یکسانی از دایرکتوری دارد؛ یک نام در تمام سرورهای LDAP به یک

ورودی ارجاع می‌کنـد. ایـن یـک ویژگـیِ بسـیار مهـم در یـک سـرویس دایرکتوری جهانی، مانند LDAP، محسوب می‌شود.

۱-۶ در مورد X.500[1]

از نظر فنی، LDAP یک پروتکل دسترسـی دایرکتـوری بـه سـرویس دایرکتوری X.500، سرویس دایرکتـوری OSI، اسـت. در ابتـدا، کـاربران LDAP بـه دروازه‌هـای سـرویس دایرکتـوری X.500 دسترسـی پیـدا می‌کنند. این دروازه LDAP را میان کاربر و دروازه و پروتکـل دسترسـی دایرکتوری X.500 (DAP) نیز LDAP را میان دروازه و سـرور X.500 اجرا می‌کند. DAP پروتکل سنگینی است که بر روی پشته پروتکل OSI کامل عمل می‌کند و نیازمند منابع پردازشی بسیار بالایی اسـت. طراحـی LDAP به‌گونه‌ای است که بر اساس TCP/IP عمل کرده و بـا هزینـه‌ای بسیار پایین، بیشتر عملکردهای DAP را ارائه می‌دهد. بـا اینکـه LDAP هنوز از طریق دروازه‌ها به X.500 متصل مـی‌شـود، امـا اکنـون LDAP معمولاً در داخل سرورهای X.500 به‌صورت مسـتقیم پیـاده‌سـازی شـده است.

LDAP daemon مسـتقل، یـا slapd(8) را مـی‌تـوان یـک سـرور دایرکتـوری X.500 سـبک‌وزن دانسـت؛ یعنـی نـه پروتکـل دسترسـی دایرکتـوری X.500 را در خـود دارد و نـه از تمـام مـدل‌هـای X.500 پشتیبانی می‌کند.

اگر در حال حاضر از سرویس پروتکـل دسترسـی دایرکتـوری X.500 استفاده می‌کنید و می‌خواهید به استفاده از آن ادامـه دهیـد، مـی‌توانیـد

---

[1] مجموعه‌ای از استانداردهای شبکه‌های کامپیوتری که سرویس‌های دایرکتوری الکترونیکی را پوشش می‌دهند.

مطالعه این راهنما را متوقف کنید. این راهنما در مورد اجـرای LDAP از طریق slapd(8)، بدون استفاده از پروتکل دسترسی دایرکتـوری X.500 است؛ اما اگر از پروتکل دسترسی دایرکتوری X.500 استفاده نمی‌کنید و یا می‌خواهید استفاده از آن را متوقف کنید و یا برنامه‌ای کوتاه مدت برای استفاده از آن ندارید، به خواندن این نوشته ادامه دهید.

replication از دایرکتـــوری LDAP بـــه X.500 DAP DSA امکان‌پذیر است. این کار نیازمنـد یـک دروازه LDPA/DAP مـی‌باشـد. OpenLDAP این دروازه را ارائه نمی‌دهد، امـا Daemon جـایگزین مـا می‌تواند برای کپی به چنـین دروازه‌ای مـورد اسـتفاده قـرار گیـرد. بـرای اطلاعات بیشتر در مـورد replication اطلاعـات، فصـل replication بـا slurpd را مطالعه نمایید.

۱-۷ LDAP در برابر RDBMS

این سؤال بارها در حالت‌های مختلف پیش می‌آید که رایج‌ترین آن‌هـا این اسـت: *چـرا OpenLDAP، ماننـد LMDB، بـه‌جـای یـک فهرسـت کلید/مقـدار ادغـام شـده، از یـک سیسـتم مـدیریت پایگـاه داده وابسـته (RDBMS) اسـتفاده نمـی‌کنـد؟* در حالـت کلـی، انتظـار مـی‌رود کـه الگوریتم‌های پیشرفته اجرا شده توسط RDBMS تجاری باعث سریع‌تـر و بهتر شدن OpenLDAP شده و هم‌زمان، اجازه به اشتراک‌گذاری داده با دیگر برنامه‌ها را فراهم آورند.

پاسخ کوتاه به این سؤال این است که استفاده از یک پایگاه داده ادغام شده و سیستم ورود سفارشی به OpenLDAP این اجازه را می‌دهـد کـه عملکرد و مقیاس‌پذیریِ بیشتری را بدون از دست رفـتن قابلیـت اعتمـاد ارائـه دهـد. OpenLDAP از نـرم‌افـزار پایگـاه داده هـم‌زمان/تراکنشـی استفاده می‌کند.

پاسخ بلند: همه ما همیشه در برابر انتخاب‌های RDBMSها در مقابل دایرکتوری‌ها قرار گرفته‌ایم. انتخاب میان این دو کار دشواری است و پاسخ آسانی برای آن وجود ندارد.

معمولاً تصور می‌شود که داشتن پایانه‌های RDBMS به دایرکتوری تمام مسائل را حل می‌کند؛ اما اصلاً این‌طور نیست. علت آن هم این است که مدل‌های داده بسیار متفاوت هستند. نمایش داده دایرکتوری توسط یک دایرکتوریِ وابسته نیازمند تقسیم داده به چندین جدول است.

لحظه‌ای به ObjectClass یک فرد فکر کنید. تعریف آن نیازمند نوع ویژگی‌های ObjectClass، sn و cn بوده و نوع ویژگی‌های userPassword، telephoneNumber، seeAlso و توصیف را نیز مجاز می‌کند. تمام این ویژگی‌ها چند مقداری[1] هستند، پس برای normalization باید هر نوع ویژگی در جدول جداگانه‌ای قرار بگیرد.

اکنون باید در مورد کلیدهای مناسب برای آن جدول‌ها تصمیم بگیرید. کلید اصلی می‌تواند ترکیبی از DN باشد، اما این در اکثر پیاده‌سازی‌های پایگاه داده غیرموثر است.

اکنون مشکل بزرگ این است که دسترسی به داده از یک ورودی نیازمند بررسی مناطق مختلفی از دیسک می‌باشد. در برخی برنامه‌ها این ممکن است مشکلی نداشته باشد، اما در بسیاری از برنامه‌ها عملکرد به مشکل برمی‌خورد.

تنها نوع ویژگی‌هایی که می‌توانند در ورودیِ جدول اصلی قرار بگیرند نوع ویژگی‌های الزامی و تک مقداری[2] هستند. همچنین می‌توانید

---

[1] Multiple-value
[2] Single-Value

ویژگی‌های تک مقداری اختیاری نیز اضافه کرده و آن‌ها را به حالت NULL و یا در صورت ناموجود بودنِ آن، به چیز دیگری تنظیم کنید.

اما صبر کنید، ورودی می‌تواند دارای ObjectClass چندگانه بوده و در سلسله‌مراتب وراثتی سازمان‌دهی شده باشد. یک ورودیِ objectClass organizationalPerson دارای ویژگی‌های فردی و چند ویژگیِ دیگر بوده که اکنون چند نوع ویژگیِ اختیاری، الزامی شده‌اند.

چه باید کرد؟ آیا باید جدول‌های مختلفی برای ObjectClassهای مختلف داشته باشیم؟ در این حالت فرد یک ورودی در جدول فرد و یک ورودی در organizationalPerson و ... خواهد داشت. یا باید از شرّ فرد خلاص شده و همه چیز را در جدول دوم قرار دهیم؟

با فیلتری مانند (cn=*) که در آن cn نوع ویژگی‌ای است که در objectCLassهای بسیاری قرار دارد چه کار کنیم؟ آیا باید در تمام جدول‌های ممکن به جست‌وجوی ورودی‌های مطابق بپردازیم؟ زیاد جذاب به نظر نمی‌رسد.

وقتی که به اینجا رسیدیم، سه رویکرد به ذهن می‌آید.

اولین رویکرد این است که یک هنجار سازی کامل انجام دهیم تا هر نوع ویژگی، هر چه می‌خواهد باشد، دارای جدول جداگانه‌ای باشد. رویکرد ساده‌ای که در آن DN بخشی از کلید اصلی است که به شدت بی‌فایده بوده و نیازمند روشی است که در آن ورودی دارای id عددی منحصر به فردی باشد که به‌جای کلیدها مورد استفاده قرار گرفته و یک جدول اصلی وجود داشته باشد که DNها را به idها مسیردهی کند. البته هنگامی که انواع ویژگی‌های مختلف از یک یا چند ورودی درخواست می‌شوند، زیاد مفید به نظر نمی‌رسد. چنین پایگاه‌داده‌ای، با اینکه بسیار سنگین است، می‌تواند توسط برنامه‌های SQL مدیریت شود.

رویکرد دوم این است که کل ورودی را به‌صورت blob در جدولی قرار دهیم که بین تمام ورودی‌ها، صرف‌نظر از objectClass، مشترک باشد و سپس جدول‌های اضافه‌ای داشته باشیم که به عنوان شاخص برای جدول اول عمل کنند. جدول‌های شاخص، شاخص‌های پایگاه داده نیستند، بلکه کاملاً توسط پیاده‌سازیِ LDAP سمت سرور مدیریت می‌شوند. با این همه، این پایگاه داده برای SQL غیرقابل استفاده می‌شود؛ بنابراین، یک سیستم پایگاه داده کاملاً تکامل یافته هیچ مزیتی ندارد. عمومیت کامل پایگاه داده مورد نیاز نیست. استفاده از سیستم سبک‌تر و سریع‌تر، مانند LMDB بسیار بهتر است.

یک روش کاملاً متفاوت برای این مسئله این است که امید خود را در مورد پیاده‌سازیِ مدل داده دایرکتوری از دست بدهیم. در این حالت، LDAP به عنوان یک پروتکل دسترسی به داده‌ها مورد استفاده قرار می‌گیرد که تنها به‌صورت مصنوعی مدل داده دایرکتوری را ارائه می‌دهد. برای مثال، ممکن است تنها قابل خواندن باشد و یا به‌روزرسانی داده‌ها مجاز باشد، محدودیت‌ها اعمال شوند، مانند ایجاد انواع ویژگیِ تک‌مقداری که مقادیر چندگانه را نیز ممکن می‌سازد؛ و یا عدم امکان اضافه کردن objectClass جدید به ورودیِ موجود و یا حذف یکی از ورودی‌های موجود ممکن باشد. بازه محدودیت‌ها از محدودیت‌های مجاز (که ممکن است در جای دیگر نتیجه کنترل دسترسی باشد) تا کاملاً نقض مدل داده می‌باشد. البته می‌تواند یک متد برای ارائه دسترسیِ LDAP به داده‌های از پیش موجود باشد که توسط دیگر برنامه‌ها مورد استفاده قرار می‌گیرد؛ اما باید این را دانست که ما در واقع یک «دایرکتوری» نداریم.

پیاده‌سازی‌های سرورهای LDAP تجاری موجود که از یک پایگاه داده وابسته استفاده می‌کنند، یا از نوع اول هستند و یا نوع سوم. هیچ

نوع پیاده‌سازی وجود ندارد که از یک پایگاه داده وابسته برای انجام غیرمفید کاری استفاده کند که BDB به‌صورت مؤثر آن را انجام می‌دهد. برای آن‌هایی که به روش سوم علاقه‌مند هستند (اعمال داده‌های موجود از RDBMS به‌صورت یک درخت LDAP، داشتن چند محدودیت در مقایسه با مدل کلاسیک LDAP و ممکن کردن همکاریِ بین برنامه‌های LDAP و SQL):

OpenLDAP شامل back-sql می‌شود، backendی که این امکان را فراهم می‌کند. back-sql از ODBC و اطلاعات اضافه‌ای در مورد ترجمه جستارهای LDAP به جستارهای SQL در اسکیمای RDBMS شما استفاده می‌کند که در نهات موجب ارائه سطوح دسترسی مختلف می‌شود: از فقط خواندن تا دسترسیِ کامل با توجه به RDBMS و اسکیمایی که استفاده می‌کند.

برای اطلاعات بیشتر در این مورد و محدودیت‌هایش، صفحه اصلیِ *slapd-sql(5)* و یا بخش Backend ها را مشاهده نمایید. همچنین در زیرشاخه‌های back-sql/rdbms_depend/* چندین مثال برای RDBMSها وجود دارد.

۱-۸ Slapd چیست و چه کاری می‌تواند انجام دهد؟

Slapd(8) یک سرور دایرکتوری LDAP است که می‌تواند در پلتفرم‌های بسیاری اجرا شود و شما می‌توانید از آن برای ایجاد یک سرویس دایرکتوری برای خودتان استفاده کنید. دایرکتوری شما می‌تواند تقریباً هر چیزی که می‌خواهید در آن قرار دهید داشته باشد. می‌توانید آن را به یک سرویس دایرکتوری LDAP جهانی متصل کنید و یا کل سرویس را به تنهایی اجرا کنید. برخی از ویژگی‌ها و قابلیت‌های جالب slapd بدین صورت است:

LDAPv3: slapd از نسخه سه پروتکل دسترسی دایرکتوری سبک‌وزن استفاده می‌کند. Slapd از LDAP هم در IPV4 و هم IPV6 و هم Unix IPC پشتیبانی می‌کند.

احراز هویت ساده و لایه امنیتی: slapd از سرویس‌های امنیت داده و احراز هویت بسیار قوی (ادغامی و محرمانه) با استفاده از SASL[1] پشتیبانی می‌کند. Slapd برای اجرای SASL از نرم‌افزار Cyrus SASL استفاده می‌کند که از چند مکانیسم از جمله DIGEST-MD5، EXTERNAL و GSSAPI پشتیبانی می‌کند.

پروتکل امنیتی لایه انتقال (TLS): slapd از احراز هویت مبتنی بر گواهی‌نامه (Certificate) و با استفاده از TLS (یا SSL)، از سرویس‌های امنیت داده (ادغامی و محرمانه) پشتیبانی می‌کند. اجرای TSL در Slapd از طریق نرم‌افزار OpenSSL، GnuTLS و MozNSS انجام می‌شود.

کنترل توپولوژی: slapd را می‌توان طوری پیکربندی کرد که بر اساس اطلاعات توپولوژی شبکه، دسترسی در لایه سوکت را محدود کند. این ویژگی با استفاده از بسته‌های TCP صورت می‌گیرد.

کنترل دسترسی: slapd امکان کنترل دسترسی پیشرفته‌ای را ارائه می‌دهد که این امکان را برای شما فراهم می‌کند تا دسترسی به اطلاعات را در پایگاه داده‌های خود کنترل کنید. شما می‌توانید بر اساس اطلاعات احراز هویت LDAP، آدرس IP، نام دامنه و دیگر معیارها، دسترسی به ورودی‌ها را کنترل کنید. slapd هم از اطلاعات پویا و هم از اطلاعات ایستا در کنترل دسترسی اطلاعات پشتیبانی می‌کند.

---

[1] چهارچوبی برای احراز هویت و امنیت داده در پروتکل‌های اینترنت.

بین‌المللی بودن: slapd از Unicode[۱] و تگ‌های زبانی پشتیبانی می‌کند.

انتخاب پایانه پایگاه داده: slapd چندین پایانه پایگاه داده را برای انتخاب شما به همراه دارد. این پایانه‌ها عبارتند از: MDB، پایگاه داده تراکنشی با عملکرد عالی و سلسله مراتبی، BDB، پایانه پایگاه داده انتقالی با عملکرد عالی، HDB، پایانه انتقال سلسله‌مراتبی با عملکرد عالی، LDBM، پایانه سبک وزن مبتنی بر DBM، SHELL، رابط پایانه‌ای برای اسکریپت پوسته اختیاری و PASSWD، رابط پایانه ساده برای فایل passwd(5). پایانه‌های DBD و HDB از Sleepycat Berkeley DB استفاده می‌کنند. LDBM یا از Berkeley DB و یا از GDBM استفاده می‌کند. MDB backend از LMDB استفاده می‌کنند. DBD و HDB با توجه به throughput بسیار بالا و چشمگیر در خواندن و نوشتن و همچنین reliability داده‌ها در LMDB تقریباً منسوخ شده‌اند.

ماژول‌های API عمومی: اگر به دنبال سفارشی‌سازیِ بیشتری هستید، slapd به شما اجازه می‌دهد که به راحتی ماژول‌های خود را بنویسید. slapd از دو بخش مجزا تشکیل شده است: قسمت جلویی (Front-end) که ارتباط پروتکل با دیگر کاربران LDAP را به عهده دارد و ماژول‌ها که کارهای جزئی مانند عملیات پایگاه داده را انجام می‌دهند. چون این دو بخش از طریق یک C API به‌خوبی تعریف شده ارتباط برقرار می‌کنند، شما می‌توانید ماژول‌های سفارشی خود را بنویسید که slapd را جنبه‌های مختلف توسعه دهند. چند ماژول پایگاه داده‌ای قابل

---

[۱] یونی‌کُد[۱] (به انگلیسی: Unicode) استانداردی صنعتی برای رمزنگاری نویسه‌های رایانه‌ای و نمایش و پردازش متن به اکثر زبان‌های دنیا[۲] است.

برنامه‌نویسی نیز ارائه می‌شود که به شما اجازه می‌دهند تا با استفاده از زبان‌ها برنامه‌نویسی مشهور (Perl، Shell، SQL و TCL) منابع داده‌ی خارجی را به slapd تحمیل کنید.

رشته‌ها: slapd برای عملکرد بالا به‌صورت رشته در آمده است. یک فرآیند ساده‌ی چند رشته‌ای تمام درخواست‌های ورودی را با استفاده از منبع رشته‌ها انجام می‌دهد. این کار باعث کاهش سربار مورد نیاز شده و عملکرد خوبی ارائه می‌دهد.

Replication[1]: slapd برای replication از دو معماری single-master/multiple-slave و multi-master پشتیبانی می‌کند. همچنین slapd از LDAP Sync-based replication نیز پشتیبانی می‌کند.

کش پروکسی: پیکربندی slapd می‌تواند به‌گونه‌ای باشد که به عنوان یک سرویس LDAP پروکسی عمل کند.

پیکربندی: از طریق یک فایل پیکربندی که اجازه تغییر هر چیزی که می‌خواهید را برایتان فراهم می‌کند، به راحتی می‌توان slapd را پیکربندی کرد. گزینه‌های پیکربندی، پیش‌فرض‌های معقول و منطقی‌ای دارند که کار شما را بسیار آسان‌تر می‌کنند. پیکربندی همچنین می‌تواند از طریق LDAP به‌صورت dynamic صورت پذیرد که به‌صورت چشمگیری مدیریت را بهبود می‌بخشد.

۱-۹ خلاصه، مثال‌ها و سناریوهای عملی اجرا شده

---

[1] تکرار یا Replication در سیستم‌های کامپیوتری یعنی اشتراک‌گذاری اطلاعات به‌صورتی که در منابع مختلف ثبات وجود داشته باشد، از جمله اجزاء سخت‌افزاری و نرم‌افزاری که هدف آن بهبود اتکاء، تلرانس خطا و قابلیت دسترسی است.