Bazy danych SQL. Teoria i praktyka

Autor: Wiesław Dudek
ISBN: 83-246-0503-7
Format: B5, stron: 4882
Przykłady na ftp: 1929 kB

Bazy danych to aplikacje, z których korzystają niemal wszyscy użytkownicy komputerów, czasem nawet nie zdając sobie z tego sprawy. W bazach danych przechowywane są informacje o użytkownikach witryny WWW, kontrahentach firmy czy numerach telefonów abonentów operatora telekomunikacyjnego. Bazą danych jest również rejestr systemu Windows i książka telefoniczna w telefonie komórkowym. Jednak aby zapisać dane w formacie odpowiednim dla aplikacji niezbędne są standardy. Współcześnie wykorzystywany relacyjny model przechowywania danych sprawdza się znakomicie. Do manipulowania danymi zapisanymi w bazach stosowany jest inny standard: język SQL.

Książka „Bazy danych SQL. Teoria i praktyka" przedstawia wszystkie zagadnienia związane z przechowywaniem i przetwarzaniem danych we współczesnych aplikacjach. Przeczytasz w niej o relacyjnym i obiektowym modelu danych oraz najczęściej stosowanych systemach zarządzania bazami danych. Dowiesz się, jakie instrukcje języka SQL wykorzystywane są do wprowadzania danych, przetwarzania ich i wybierania z bazy. Nauczysz się optymalizować zapytania oraz stosować indeksy i procedury składowane. W książce znajdziesz również praktyczne wskazówki dotyczące konfigurowania serwerów baz danych i administrowania nimi.

- Przechowywanie prostych danych
- Relacyjny i obiektowy model danych
- Typy danych w języku SQL
- Instrukcje języka SQL
- Indeksy, wyzwalacze i procedury składowane
- Manipulowanie danymi
- Optymalizacja zapytań
- Charakterystyka najpopularniejszych systemów zarządzania bazami danych

Poznaj tajniki współczesnych mechanizmów przechowywania informacji

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 3.

LDAP — hierarchiczna baza danych

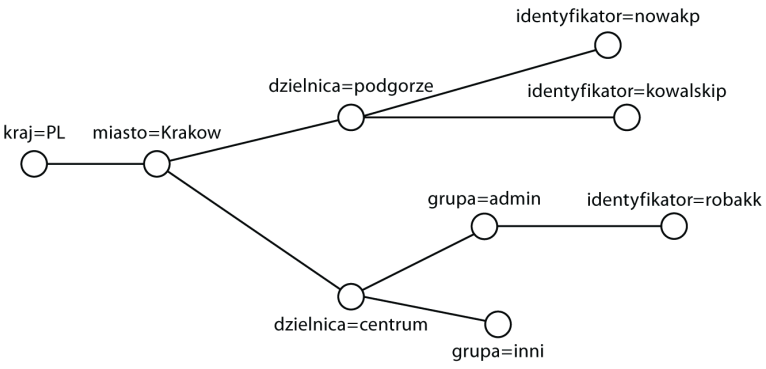
Opis technologii LDAP prezentowany w tym rozdziale będzie się głównie opierał na dystrybucji serwera OpenLDAP w wersji 2.x (*www.OpenLDAP.org*).

Krótka charakterystyka bazy

Z pojęciem LDAP wiąże się baza danych oparta na hierarchicznej strukturze, o typie danych atrybut-wartość oraz protokole dostępu działającym w oparciu o TCP/IP.

Hierarchia bazy danych przypomina drzewo katalogu plików (rysunek 3.1), gdzie DN (*distinguish name*) odpowiada ścieżce dostępu do zbioru atrybutów i identyfikuje węzeł drzewa (*entry*).

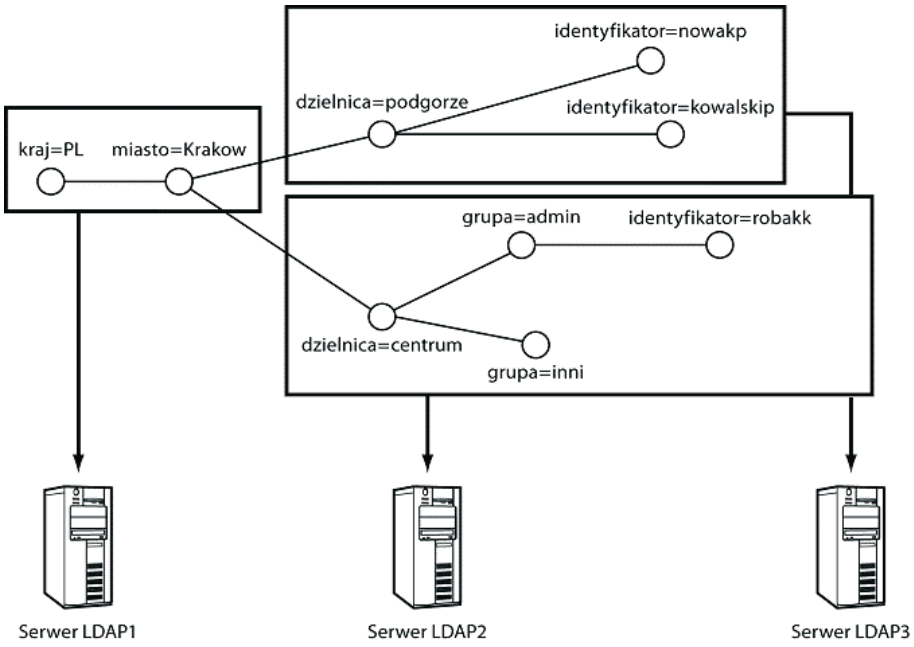
Każdy taki węzeł posiada skojarzony z nim zbiór obiektów, które jednoznacznie definiują zbiór jego dozwolonych atrybutów. Obiekty i atrybuty dla bazy danych LDAP definiuje schemat bazy danych (patrz: punkt „Schemat").



Rysunek 3.1. *Hierarchiczna struktura bazy LDAP*

Założeniem LDAP jest uzyskanie prostoty dostępu do danych oraz niewielka ilość operacji zapisu w porównaniu z operacjami odczytu obserwowana w rzeczywistych systemach. W wyniku tych założeń LDAP nie oferuje transakcji.

Serwer LDAP może być skonfigurowany tak, aby delegował odpowiedzialność za podgałąź do innego serwera LDAP działającego zwykle na innym komputerze (przedstawia to rysunek 3.2). Dzięki takiemu rozwiązaniu bazy LDAP charakteryzują się dużą skalowalnością.



Rysunek 3.2. *Przydział węzłów do serwerów LDAP*

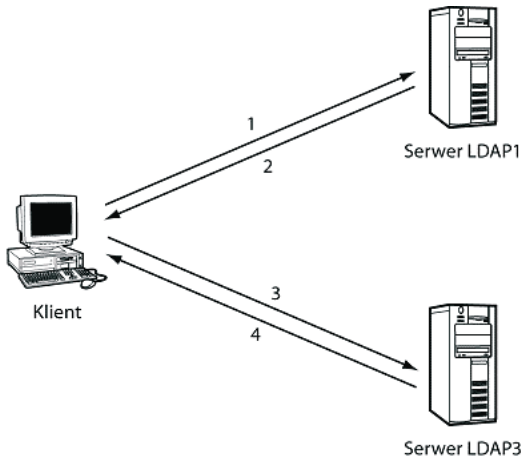
W systemie skonfigurowanym jak na rysunku 3.2 każde żądanie dostępu do węzła o DN: `dzielnica=podgorze`, `miasto=Krakow`, `kraj=PL`, które zostanie zgłoszone do serwera LDAP1, zostanie przekierowane do serwera LDAP3, który został oddelegowany do obsługi podgałęzi `"dzielnica=podgorze,miasto=Krakow,kraj=PL"`. Poszczególne etapy obsługi żądania przedstawia rysunek 3.3.

Obsługa przekierowania powinna zostać zaimplementowana przez program klienta. Istnieje jednak bardzo wiele programów klienckich nieobsługujących przekierowań i ignorujących informację otrzymaną od serwera LDAP1.

LDIF

LDIF (*LDAP Data Intercharge Format*) jest formatem tekstowym reprezentującym dane LDAP. Ponieważ użyto formatu tekstowego, więc dane te mogą być z łatwością edytowane przez użytkownika.

Rysunek 3.3. *Obsługa żądania dostępu do oddelegowanego węzła*



1. Żądanie dostępu do atrybutu węzła DN: kraj=PL,miasto=Krakow,dzielnica=podgorze
2. Serwer LDAP1 przekierowuje klienta do serwera LDAP3
3. Klient ponawia żądanie zgłoszając je serwerowi LDAP3
4. Serwer LDAP3 zwraca wynik

W uproszczeniu format pliku LDIF przedstawia się następująco:

```
# komentarz
dn: <nazwa jednoznaczna>
<atrybut>: <wartość>
<atrybut>: <wartość>
```

Pliki LDIF mogą jednak posiadać bogatszą strukturę. Informacje na ten temat można znaleźć w dokumencie RFC 2849 [1].

Przykład pliku LDIF:

```
dn: identyfikator=kowalj,miasto=krakow,kraj=pl
objectclass: osoba
identyfikator: kowalj
nazwisko: kowal
imie: jan
wiek: 20
```

Tworząc pliki LDIF, możemy łatwo przygotować bazę danych dla serwera LDAP, a następnie skorzystać z narzędzia `ldiff2ldbm`, które potrafi utworzyć na jego podstawie bazę *ldbm*.

```
ldif2ldbm -f slapd.conf -i my.ldif
```

Odwrotną operację konwersji bazy danych *ldbm* na format LDIF umożliwia `ldbmcat`.

```
ldbmcat -n id2entry.dbb > my.ldif
```

Jednak nie możemy użyć narzędzia `ldif2ldbm`, jeśli jako bazy danych dla serwera LDAP (tzw. *backend*) używamy bazy innej niż *ldbm*. W tej sytuacji mamy ciągle do dyspozycji standardowe narzędzia dostarczane z serwerem i opisane w punkcie „Narzędzia takie jak: `ldapadd`, `ldapmodify`, `ldapdelete` itd.”.

Utworzenie bazy danych na podstawie pliku LDIF za pomocą `ldapadd` przedstawia się następująco:

```
ldapadd -v -D "identyfikator=manager,miasto=krakow,kraj=PL" -w manager -f baza.ldif
```

Jeśli najpierw zatrzymamy serwer, możemy posłużyć się również narzędziem `slapadd`:

```
slapadd -l baza.ldif -cv
```

Schemat

Schemat LDAP zawiera definicje bazy danych: typy przechowywanych danych, dozwolone wartości lub zakresy atrybutów, wymagalność lub opcjonalność atrybutów, informacje o więzach narzuconych na przechowywane wartości, takich jak brak duplikatów, czy informacje o sposobie porównywania wartości. Schemat jest więc definicją struktury bazy.

Schematy są odczytywane podczas startu serwera i tylko dane zgodne z regułami zawartymi w schemacie są dozwolone w bazie danych LDAP.

Z serwerem OpenLDAP dostarczanych jest kilka plików schematów, które mogą okazać się użyteczne dla użytkownika. Jedynie schemat *core.schema* jest wymagany do pracy serwera LDAP, pozostałe są opcjonalne.

W dostarczonych z serwerem OpenLDAP plikach schematów zdefiniowano wiele obiektów i atrybutów, które mogą być wykorzystane przy tworzeniu własnej bazy danych. Przykładowo tabela 3.1 zawiera listę atrybutów wraz z odpowiadającymi im obiektami, które mogą okazać się przydatne podczas tworzenia własnej bazy użytkowników.

Oczywiście użytkownik może w miarę swoich potrzeb tworzyć własne pliki schematów, które może dołączyć do pliku konfiguracyjnego *sldap.conf* za pomocą instrukcji `include`.

Przykład prostego pliku schematu przedstawia listing 3.1:

Listing 3.1. *Przykładowy plik schematu*

```
# Autor: Wiesław Dudek
# definicja typów atrybutów

attributetype ( 1.3.6.1.4.1.4203.666.1 NAME 'identyfikator'
        EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE)

attributetype ( 1.3.6.1.4.1.4203.666.2 NAME 'nazwisko'
        EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE)

attributetype ( 1.3.6.1.4.1.4203.666.3 NAME 'kraj'
        EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE)
```

Tabela 3.1. *Lista ważniejszych atrybutów zdefiniowanych w schematach dostarczonych wraz z serwerem OpenLDAP*

Atrybut	Nazwa pola w MSOutlook	Plik schematu z definicją	Nazwa obiektu, w którym występuje atrybut
cn	*Name:*	*core.schema*	objectPerson
givenName	*First Name:*	*core.schema*	inetOrgPerson
initials	*Middle Name:*	*core.schema*	inetOrgPerson
sn	*Last Name:*	*core.schema*	objectPerson
mail	*Email Address:*	*core.schema*	inetOrgPerson
title	*Job Title:*	*core.schema*	organizationalPerson
physicalDeliveryOfficeName	*Office:*	*core.schema*	organizationalPerson
o	*Company Name:*	*core.schema*	inetOrgPerson
postalAddress	*Business — Street Address:*	*core.schema*	organizationalPerson
l	*Business — City:*	*core.schema*	organizationalPerson
st	*Business — State/Province:*	*core.schema*	organizationalPerson
postalCode	*Business — Zip Code:*	*core.schema*	organizationalPerson
c	*Business — Country/Region:*	*core.schema*	officePerson
telephoneNumber	*Business:*	*core.schema*	organizationalPerson
facsimileTelephoneNumber	*Business Fax:*	*core.schema*	organizationalPerson
homePhone	*Home:*	*cosine.schema*	inetOrgPerson
mobile	*Mobile:*	*cosine.schema*	inetOrgPerson
homePostalAddress	*Home — Street Address:*	*cosine.schema*	inetOrgPerson
manager	*Manager:*	*cosine.schema*	inetOrgPerson
pager	*Pager:*	*cosine.schema*	inetOrgPerson

```
attributetype ( 1.3.6.1.4.1.4203.666.4 NAME 'miasto'
        EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE)

attributetype ( 1.3.6.1.4.1.4203.666.5 NAME 'dzielnica'
        EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE)

attributetype ( 1.3.6.1.4.1.4203.666.6 NAME 'imie'
        DESC 'Imiona'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

```
attributetype ( 1.3.6.1.4.1.4203.666.7 NAME 'wiek'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE)

#definicja typów obiektów

objectclass ( 1.3.6.1.4.1.4203.666.8 NAME 'osoba'
  MUST ( identyfikator $ nazwisko $ imie)
  MAY ( wiek ) )

objectclass (1.3.6.1.4.1.4203.666.9 NAME 'miasteczko'
  MUST ( miasto ))

objectclass ( 1.3.6.1.4.1.4203.666.10 NAME 'rejon'
  MUST ( dzielnica ))
```

Jak widzimy, każdy element schematu posiada własny identyfikator obiektu, tzw. *OID* (*Object Identifier*), np. *OID* elementu `"rejon"` to `1.3.6.1.4.1.4203.666.10`. Numer ten rozpoczyna się od prefiksu przyznanego organizacji przez właściwy urząd (np. przez urząd IANA[2]). W przykładzie użyliśmy prefiksu `1.3.6.1.4.1.4203.666`, który jest zarezerwowany przez *OpenLDAP.org* dla celów „eksperymentalnych". Jednak mogliśmy użyć dowolnego prefiksu zgodnego z notacją OID niekolidującego z prefiksami użytymi w innych używanych przez nas schematach, ponieważ nasz plik schematu jest przeznaczony tylko do własnego użytku. Kolejne elementy schematu otrzymują numery rozpoczynające się od prefiksu: 1.3.6.1.4.1.4203.666.1, 1.3.6.1.4.1.4203.666.2 itp.

W naszym przykładowym pliku schematu pojawiły się dwa typy definicji: definicje typów atrybutów rozpoczynające się od słowa `attributetype` oraz definicje typów obiektów rozpoczynające się od słowa `objectclass`. Definicja typu obiektu określa, jakie atrybuty muszą lub mogą stać się jego składnikami.

Składowe definicji typu atrybutu

`NAME` — nazwa atrybutu. Nazwa ta, podobnie jak numer OID, powinna być unikalna. Jeśli schemat będzie publicznie wykorzystywany, dobrym pomysłem jest poprzedzanie nazw naszych atrybutów prefiksem organizacji, podobnie jak to ma miejsce w przypadku numeru OID.

Dozwolone jest użycie kilku nazw dla atrybutu, np. `attributetype ( 2.5.4.7 NAME ( 'l' 'localityName' ) SUP name )`.

`DESC` — opis atrybutu.

`SUP` — deklaracja dziedziczenia; definiowanie nowego atrybutu na podstawie istniejącego.

Przykład pochodzący z pliku *core.schema* — definiowanie atrybutu `member`:

```
attributetype ( 2.5.4.31 NAME 'member' SUP distinguishedName )
```

Atrybut `member` dziedziczy właściwości atrybutu `distinguishedName`.

`EQUALITY` — określa regułę porównywania elementów (patrz: tabela 3.2).

`ORDERING` — określa regułę porównywania elementów za pomocą operatorów <= i >= (możliwe wartości patrz: tabela 3.2).

Tabela 3.2. *Reguły dopasowywania*

Nazwa	Stosowana do	Znaczenie
`booleanMatch`	EQUALITY	porównywanie wartości logicznych
`caseIgnoreMatch`	EQUALITY	brak wrażliwości na wielkość liter i odstępy
`caseExactMatch`	EQUALITY	wrażliwe na wielkość liter, niewrażliwe na odstępy
`distinguishedNameMatch`	EQUALITY	porównywanie wartości DN
`integerMatch`	EQUALITY	porównywanie liczb całkowitych
`numericStringMatch`	EQUALITY	porównywanie ciągów numerycznych
`octetStringMatch`	EQUALITY	porównywanie ciągów bajtów
`objectIdentiferMatch`	EQUALITY	porównywanie OID-ów
`caseIgnoreOrderingMatch`	ORDERING	niewrażliwe na wielkość liter i odstępy
`caseExactOrderingMatch`	ORDERING	wrażliwe na wielkość liter, niewrażliwe na odstępy
`integerOrderingMatch`	ORDERING	porównywanie liczb całkowitych
`numericStringOrderingMatch`	ORDERING	porównywanie ciągów numerycznych
`octetStringOrderingStringMatch`	ORDERING	porównywanie ciągów bajtów
`octetStringSubstringsStringMatch`	ORDERING	porównywanie ciągów bajtów
`caseIgnoreSubstringsMatch`	SUBSTR	niewrażliwe na wielkość liter i odstępy
`caseExactSubstringsMatch`	SUBSTR	wrażliwe na wielkość liter, niewrażliwe na odstępy
`numericStringSubstringsMatch`	SUBSTR	porównywanie ciągów numerycznych

`SUBSTR` — określa regułę porównywania elementów za pomocą znaków wieloznacznych (wildcards) (patrz: tabela 3.2).

`SYNTAX` — typ atrybutu (najczęściej wykorzystywane typy atrybutów patrz: tabela 3.3).

Przykład deklaracji:

`SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{30}` określa atrybut typu *DirectoryString* kodowany w UTF-8 o długości maksymalnej 30 znaków.

`SINGLE-VALUE` — blokuje wstawianie duplikatów.

`NO-USER-MODIFICATION` — blokuje możliwość modyfikacji atrybutu.

`COLLECTIVE` — atrybut będzie definiował kolekcję węzłów.

`USAGE` — jedna z wartości: `userApplications`, `directoryOperation`, `distributedOperation`, `dSAOperation`.

Tabela 3.3. *Typy atrybutów (SYNTAX)*

Nazwa	OID	Znaczenie
boolean	1.3.6.1.4.1.1466.115.121.1.7	wartość logiczna
DirectoryString	1.3.6.1.4.1.1466.115.121.1.15	ciąg znaków kodowany w UTF-8 ***Można zapisywać polskie znaki.***
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	LDAP DN
integer	1.3.6.1.4.1.1466.115.121.1.27	liczba całkowita
numericString	1.3.6.1.4.1.1466.115.121.1.36	ciąg znaków numerycznych
OID	1.3.6.1.4.1.1466.115.121.1.38	identyfikator obiektu
octetString	1.3.6.1.4.1.1466.115.121.1.40	ciąg bajtów, np. rysunek lub inny obiekt binarny kodowany w standardzie Base64

Składowe definicji typu obiektu

NAME — nazwa obiektu. Nazwa ta powinna być unikalna.

DESC — opis obiektu.

SUP — deklarowanie dziedziczenia (określanie nowego obiektu na podstawie istniejącego).

ABSTRACT — deklarowanie obiektu abstrakcyjnego, tj. będącego punktem wyjścia do definiowania nowych obiektów.

MUST — deklarowanie atrybutów, które muszą towarzyszyć obiektowi.

MAY — deklarowanie atrybutów, które mogą (ale nie muszą) towarzyszyć obiektowi.

Zalety i wady

W podrozdziałach „Zalety i wady" standardowo występujących w każdym z następnych rozdziałów zostaną przedstawione wybrane cechy opisywanego systemu, które z różnych powodów zasługują na uwagę. Trzeba jednak pamiętać, że sytuacja na rynku informatycznym zmienia się z dnia na dzień i opisywane wady mogą zostać usunięte w kolejnych wersjach, a zalety staną się standardem na rynku tego typu oprogramowania. Przedstawione w tej książce zestawienia mają jednak dać Czytelnikowi wyobrażenie na temat bieżącej sytuacji. Sam użytkownik zdecyduje, które z tych cech mają dla niego znaczenie. Być może pewna wada zdyskwalifikuje produkt pod kątem jakiegoś wykorzystania. Z drugiej strony użytkownik może chcieć skorzystać z pewnej opcji dostępnej w danym produkcie, która z pozoru wydawałaby się mało istotna.

Śledząc zapotrzebowanie na pewne rozwiązania, wydaje się, że stworzenie listy wad i zalet ma sens. Użytkownik może je traktować jako pewien głos w dyskusji.

Przedstawiając w dużym skrócie i z pewnością nie wyczerpując tematu, można wymienić zalety i wady technologii LDAP.

Zalety:

- małe wymagania sprzętowe;
- łatwość integracji z innym oprogramowaniem (serwerami SQL, programami pocztowymi itp.);
- bardzo dobra skalowalność;
- prostota w użyciu.

Wady:

- mała szybkość i możliwości w porównaniu z bazami danych SQL;
- brak transakcji.

Podsumowanie

Zapytania do bazy danych SQL są dużo szybsze niż odpowiadające im zapytania do bazy LDAP. Co więcej, efektywność LDAP może być nawet kilkadziesiąt razy mniejsza niż bazy danych SQL. Wybór rozwiązania może więc wydawać się bardzo prosty, jednak należy pamiętać, że bazy LDAP mają też sporo zalet. Zalety te powodują, że ciągle są one obiektem zainteresowania i stają się standardem dla pewnych rozwiązań, np.: systemu logowania użytkowników, systemu książek adresowych, informacji o użytkownikach itd. Do zalet tych należy głównie prostota i łatwość integracji z innym oprogramowaniem. Otwarty protokół LDAP i współpraca serwerów LDAP z innym oprogramowaniem, w szczególności z bazami danych SQL, może pomóc w integracji systemów będących w dyspozycji użytkownika.

Instalacja i konfiguracja

Instalacja serwera OpenLDAP jest stosunkowo prosta. Z pewnością będzie wymagać trochę więcej wysiłku w systemie Linux, zwłaszcza jeśli zechcemy skonfigurować bezpieczne uwierzytelnienie np. przez SSL.

OpenLDAP jest darmowym oprogramowaniem, które można pobrać ze strony *www.openldap.org*.

Instalacja serwera OpenLDAP zarówno w systemie Windows, jak i w systemie Linux przebiega bardzo podobnie i składa się z następujących etapów:

1. Zainstalowanie oprogramowania serwera.
2. Utworzenie plików schematu bazy danych (patrz: punkt „Schemat”).
3. Skonfigurowanie serwera polegające na modyfikacji pliku *slapd.conf* (patrz: punkt: „Plik slapd.conf”).

4. Utworzenie pliku LDIF z danymi naszej nowej bazy danych (patrz: punkt „LDIF").
5. Uruchomienie serwera LDAP (patrz: punkt „Uruchamianie i zatrzymywanie serwera").
6. Utworzenie bazy danych na podstawie pliku LDIF (patrz: punkt „LDIF").

Standardowe porty, na których nasłuchuje serwer LDAP, to port 389 dla połączenia nieszyfrowanego oraz 636 dla połączeń szyfrowanych TLS/SSL.

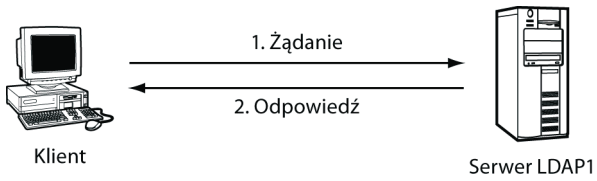
Popularne konfiguracje serwera LDAP

Najpopularniejsze konfiguracje serwera LDAP przedstawiono na kolejnych rysunkach. Niektóre z nich pozwalają na rozłożenie obciążenia na kilka serwerów (patrz: rysunek 3.5 i rysunek 3.6), celem innych jest dodatkowo uzyskanie bezpieczeństwa danych (patrz: rysunek 3.6).

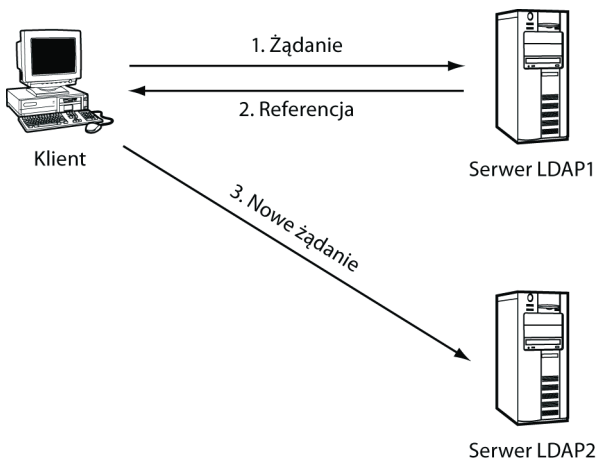
Konfiguracja podstawowa

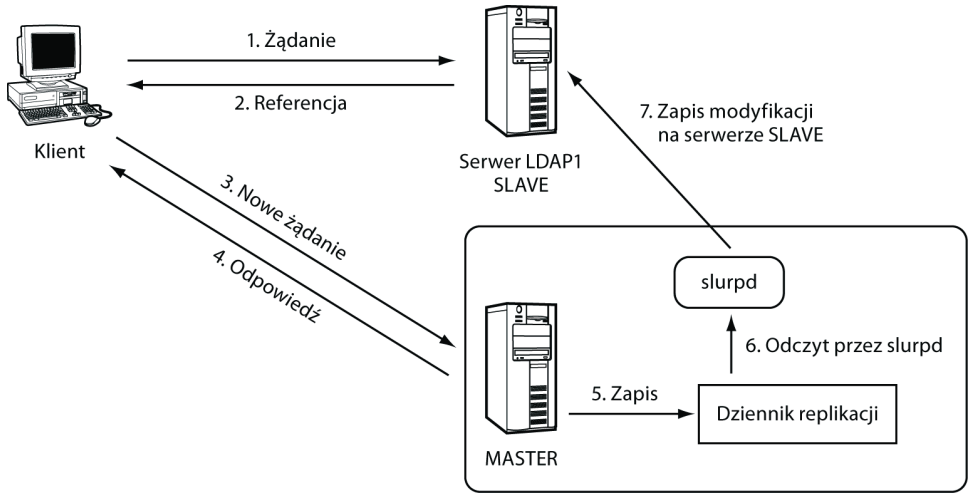
Jest to konfiguracja domyślna tworzona podczas instalacji serwera LDAP, na którą składa się jeden serwer (patrz: rysunek 3.4). Żądania nadchodzące od strony klienta są całkowicie przetwarzane na tym serwerze. Jeśli serwer nie potrafi obsłużyć żądań, informuje o tym i nie odsyła klienta do innych serwerów LDAP.

Rysunek 3.4. *Konfiguracja podstawowa*



Rysunek 3.5. *Konfiguracja z serwerami referencyjnymi*





Rysunek 3.6. *Konfiguracja z replikacją typu MASTER-SLAVE*

Konfiguracja z serwerami referencyjnymi

Ta konfiguracja pozwala uzyskać sieć powiązanych ze sobą serwerów LDAP, które odsyłają klientów do siebie nawzajem w zależności od możliwości obsłużenia ich żądań.

Na rysunku 3.5 klient w kroku 2. otrzymuje od serwera LDAP1 referencję do serwera LDAP2, który zdaniem serwera LDAP1 jest w stanie obsłużyć żądanie klienta. Klient po otrzymaniu referencji ponawia żądanie, kierując je już do właściwego serwera.

Serwery LDAP obsługują dwa rodzaje referencji:

1. Referencje definiowane w pliku *slapd.conf* za pomocą słowa `referral`

   ```
   referral  ldap://root.openldap.org/
   ```

   Jeśli żądanie klienta dotyczy węzła spoza obsługiwanej domeny określonej poprzez słowo `suffix`, serwer zwraca klientowi referencję do właściwego serwera LDAP (*ldap://root.openldap.org*), który może obsłużyć żądanie klienta lub odesłać go do innego serwera.

   Na przykład: jeśli w pliku *slapd.conf* określono

   ```
   suffix miasto=krakow,kraj=pl
   ```

   żądanie klienta dotyczące węzła `miasto=warszawa,kraj=pl` zwróci referencję zdefiniowaną w `referral`.

2. Referencje definiowane dla węzła poprzez przypisanie węzłowi obiektu `referral`

   ```
   dn: dc=poddrzewo,dc=myserver,dc=pl
   objectClass: referral
   objectClass: extensibleObject
   dc: poddrzewo
   ref: ldap://myserver.pl/dc=poddrzewo,dc=myserver,dc=pl
   ```

Każde odwołanie do podgałęzi `dc=poddrzewo,dc=myserver,dc=pl` zwróci referencję: `ldap://myserver.pl/dc=poddrzewo,dc=myserver,dc=pl`. W ten sposób obsługa tej podgałęzi zostanie oddelegowana do serwera `myserver.pl`.

Niestety wiele narzędzi klienta nie potrafi zinterpretować otrzymanej od serwera LDAP referencji i ponowić żądania z wykorzystaniem otrzymanej referencji.

Konfiguracja z replikacją typu MASTER-SLAVE

Konfiguracja ta pozwala zmniejszyć ryzyko utraty danych w przypadku awarii i poprawić bezpieczeństwo systemu, ponieważ dane są replikowane do innych baz danych LDAP. Dodatkową korzyścią uzyskiwaną tutaj jest rozproszenie danych, które podobnie jak w poprzednio omówionym przypadku pozwala na rozłożenie obciążenia serwerów. Przykładowo: serwer LDAP2 może obsługiwać jedynie modyfikacje danych, natomiast serwer LDAP1 (replika) służyć do odczytywania i przeglądania danych.

Konfiguracja ta jest oparta na architekturze *MASTER-SLAVE*. *MASTER* jest serwerem głównym, a modyfikacje jego bazy danych są propagowane dalej i zapisywane do baz danych wszystkich skonfigurowanych serwerów *SLAVE*.

Przykład żądania zapisu danych do bazy serwera LDAP1 przedstawia rysunek 3.6. Serwer LDAP1, który jest serwerem SLAVE, informuje klienta, że takie zmiany można wprowadzać jedynie na serwerze MASTER, zwracając odpowiednią referencję (2). Klient formułuje nowe żądanie, wysyłając je tym razem pod właściwy adres (3) i uzyskuje potwierdzenie (4). Zmiany wprowadzone w bazie danych serwera MASTER są zapisywane do dziennika replikacji (5) i od tego momentu są dostępne dla programu *slurpd* (6), który propaguje je dalej (7) do serwera SLAVE (lub wielu serwerów SLAVE).

Przedstawiony scenariusz opisuje próbę zapisu na serwerze SLAVE. W przypadku próby odczytu scenariusz zakończyłby się w punkcie 2. po zwróceniu odpowiedzi przez serwer SLAVE.

Dalsze informacje na temat konfigurowania replikacji są opisane w punkcie „Replikacja".

Konfiguracja mieszana

Konfiguracja mieszana stanowi połączenie konfiguracji z serwerami referencyjnymi oraz konfiguracji z replikacją typu MASTER-SLAVE i jest podstawą budowy dużych systemów.

Plik slapd.conf

Konfiguracja serwera LDAP jest oparta na pliku *slapd.conf*. Plik ten decyduje o wszystkich parametrach pracy serwera (patrz: tabela 3.4).

Przykład minimalnej konfiguracji (plik *slapd.conf*) przedstawia listing 3.2.

Listing 3.2. *Przykładowy plik slapd.conf*

```
include         ./schema/core.schema
include         ./schema/my.schema
#definicja bazy
database ldbm
suffix          "miasto=krakow,kraj=pl"
rootdn          "identyfikator=manager,miasto=krakow,kraj=pl"
rootpw          manager
directory       ./data
access to * by * write
```

Tabela 3.4. *Lista i znaczenie parametrów konfiguracyjnych pliku slapd.conf*

Parametr	Znaczenie	Użycie
`access`	Określa prawa dostępu do węzłów i atrybutów.	Patrz: punkt „Prawa dostępu do serwera”.
`defaultaccess`	Domyślny tryb dostępu, jeśli tryb dostępu nie był określony w parametrze **access**.	`defaultaccess none` — brak dostępu. `defaultaccess read` — domyślny tryb odczytu.
`include`	Włączenie pliku (plik powinien posiadać taki sam format jak *slapd.conf*).	`include core.schema` — włączenie pliku schematu.
`loglevel`	Poziom logowania.	`loglevel 255` — szczegółowy poziom logowania informacji.
`referral`	Referencja zwracana klientowi w przypadku żądań dotyczących węzłów nieobsługiwanych przez serwer.	`referral myserver.com`
`sizelimit`	Maksymalna ilość elementów zwrócona przez serwer w jednej operacji odczytu.	`sizelimit 100`
`timelimit`	Maksymalny czas (w sekundach) na obsługę żądania klienta.	`timelimit 300`

`database`	Wybrana baza danych dla serwera LDAP.
	Wszystkie parametry pojawiające się do następnego użycia parametru database dotyczą tej bazy danych (patrz: przykład slapd.conf w punkcie „Interfejs”).
		`database ldbm`
		`database shell`
		`database passwd`

`readonly`	Włącza lub wyłącza tryb *read-only*.	`readonly on`
`replica`	Informacja o powiązanym serwerze replikacji SLAVE. Patrz: punkt „Replikacja”.	`replica uri=ldap://myserver:389 binddn="identyfikator=manager,` `miasto=krakow,kraj=pl" bindmethod= simple credentials=manager`
`replogfile`	Nazwa pliku dziennika logowania zmian w bazie danych, plik ten jest używany w procesie replikacji przez program *slurpd*.	`replogfile ./logs/master-ldap.replog`

Tabela 3.4. *Lista i znaczenie parametrów konfiguracyjnych pliku slapd.conf (ciąg dalszy)*

Parametr	Znaczenie	Użycie
`rootdn`	DN logowania do serwera, (używane w opcji `-D` narzędzi takich jak *ldapadd*).	`rootdn "identyfikator=manager, miasto=krakow,kraj=pl"`
`rootpw`	Hasło logowania do serwera.	`rootpw manager`
`suffix`	Określa węzeł podstawowy dla danej bazy danych, wszystkie zapytania kierowane do tego węzła lub podwęzłów będą dotyczyły bazy, dla której jest ustawiony **suffix**.	`suffix "miasto=krakow,kraj=pl"`
`updatedn`	Określa DN uprawione do wprowadzenia zmian do repliki (SLAVE), powinno odpowiadać klauzuli **binddn** dyrektywy **replica** dla serwera MASTER. ***Opcja dostępna tylko dla SLAVE.***	`updatedn "identyfikator=manager, miasto=krakow,kraj=pl"`
`updateref`	Referencja do serwera MASTER. ***Opcja dostępna tylko dla SLAVE.***	`updateref "myserver.net"`
`directory`	Kartoteka dla bazy danych.	`directory ./data`
`index`	Definicje indeksów. Dozwolone wartości: **none** — brak indeksu, **eq** — wyszukiwanie w indeksie będzie oparte na porównaniu całych elementów, **sub** — wyszukiwanie podstringów, **pres** — wyszukiwanie przez (`st=*`), tzn. wszystkich elementów.	`index objectClass eq` — indeks dla obiektu objectClass.
`objectclass`	Patrz: punkt „Schemat”.	
`attributetype`	Patrz: punkt „Schemat”.	
`schemacheck`	Włączenie lub wyłączenie sprawdzenia pliku schematu.	`schemacheck on`
`TLSCipherSuite`	Akceptowalny sposób szyfrowania.	`TLSCipherSuite HIGH:MEDIUM:+SSLv2`
`TLSCACertificateFile`	Plik z certyfikatem.	`TLSCACertificateFile cacert.pem`
`TLSCACertificatePath`	Kartoteka zawierająca certyfikaty CA.	`TLSCACertificatePath ./cert`
`TLSCertificateFile`	Plik z certyfikatem serwera.	`TLSCertificateFile servercrt.pem`
`TLSCertificateKeyFile`	Plik klucza prywatnego serwera.	`TLSCertificateKeyFile serverkey.pem`
`TLSVerifyClient`	Poziom uwierzytelnienia klienta: **never**, **allow**, **try**, **demand**, **hard**.	`TLSVerifyClient demand`

Instalacja książki adresowej LDAP

Instalacja książki adresowej LDAP w programie Microsoft Outlook Express przedstawia się następująco:

1. W menu *Narzędzia* należy wybrać *Konta*.
2. W oknie *Konta internetowe* po wybraniu *Dodaj* wybieramy *Usługa katalogowa*.
3. W oknie *Kreator połączeń internetowych* należy podać URL serwera LDAP, np. *ldap.myserver.pl* lub *localhost*, jeśli nasz serwer uruchomiliśmy na lokalnym komputerze.
4. Ustalamy, czy musimy logować się do serwera.
5. Jeśli wybraliśmy opcję logowania do serwera, powinniśmy podać DN logowania i hasło, np.:

```
Nazwa konta: identyfikator=manager,miasto=krakow,kraj=pl
Hasło: manager
```

6. Decydujemy, czy program MS Outlook Express powinien sprawdzać wpisywane adresy e-mail w bazie danych LDAP.
7. W oknie *Konta internetowe* w zakładce *Usługa katalogowa* wybieramy z listy skonfigurowany przez nas serwer i naciskamy przycisk *Właściwości*.
8. Wybieramy zakładkę *Zaawansowane* i sprawdzamy, czy ustawienie portu jest właściwe (jest to szczególnie ważne, jeśli serwera nie uruchomiliśmy na standardowym numerze portu) i jeśli jest niewłaściwe, poprawiamy je. Wprowadzamy również poprawnie *Bazę wyszukiwania*, czyli węzeł stanowiący podstawę przeszukiwania drzewa LDAP.
9. Naciskamy *OK*.

Aby skorzystać z tak skonfigurowanej książki adresowej:

1. Otwieramy książkę adresową, naciskając przycisk: *Adresy*.
2. W menu *Edycja* wybieramy *Znajdź osoby*.
3. Na liście *Szukaj w* wybieramy skonfigurowany przez nas serwer LDAP.
4. Wprowadzamy np. nazwę użytkownika w polu *Nazwa* (może być *) i naciskamy *OK*.