Stosowanie języków narodowych w rozległych systemach baz danych na podstawie X.500 Directory

Maria Górecka
Tomasz M. Wolniewicz

1995–07–18

1 Wstęp

Niniejszy raport ma na celu przedstawienie wybranej problematyki stosowania języków narodowych w rozległych, rozproszonych bazach danych. Główny nacisk kładziemy na funcjonowanie bazy X.500 Directory na przykładzie konkretnej implementacji standardu, z jaką mamy w praktyce do czynienia i konieczności dostosowywania działań do istniejących realiów.

Sporo miejsca poświęcamy na przedstawienie formalnych aspektów standardu X.500. Wydaje się nam to niezbędne dla szczegółowego wyjaśnienia sposobów reprezentacji informacji w omawianej bazie.

Część przedstawionej pracy została wykonana w ramach przygotowań do ładowania polskiej bazy danych X.500 i sfinansowana z grantu KBN nr 910/OBL.

2 Baza X.500 i jej międzynarodowa specyfika

Terminem X.500 określa się potocznie funkcjonującą jako usługa sieci Internet, rozproszoną bazę danych o instytucjach i ich pracownikach. Faktycznie X.500 jest standardem OSI, a funkcjonująca usługa jedynie jego implementacją. Terminu X.500 będziemy używać zarówno w potocznym jak i w formalnym znaczeniu, z kontekstu będzie jasne, o jakie znaczenie chodzi.

Jako standard X.500 definiuje obiektową, rozproszoną bazę danych opartą na modelu sieciowym. Pomimo, że podstawowym założeniem przyświecającym twórcom standardu było opracowanie światowej „książki adresowej", X.500 dopuszcza przechowywanie w bazie danych najzupełniej dowolnych typów.

Standard X.500 został opublikowany w 1988 roku, a pod koniec roku 1994 oczekiwana jest publikacja nowej jego edycji. Zmieniony standard (nazywany X.500 — wersja 1993) znacznie rozszerza wersję z roku 1988, zmienia się między innymi sposób definiowania reguł syntaktycznych atrybutów, jak i sposobu przyporządkowywania identyfikatorów obiektom bazy.

W niniejszym raporcie bazujemy na wersji z roku 1988 i jej rozszerzeniach wprowadzonych przez implementację QUIPU. Należy oczekiwać, że jeszcze przez dłuższy czas dominować będą implementacje tej właśnie wersji standardu. Przewidujemy, ze po opublikowaniu nowej wersji niezbędna będzie konfrontacja naszych przemyśleń i wyników z nowymi ustaleniami.

Podstawowym założeniem X.500 jest rozproszenie systemu i ogromny docelowy rozmiar bazy. W pełni użyteczny system oparty o X.500 powinien posiadać tysiące węzłów w każdym kraju i w skali świata miliardy obiektów.

Jak wspomnieliśmy powyżej X.500 zdefinowano przede wszystkim dla celów szeroko rozumianej informacji adresowej. Fakt ten ma istotny wpływ zarówno na sam standard, jego implementacje jak i funkcjonujące systemy. Wynika stąd oczywiście konieczność takiego zapisywania danych w bazie, aby z jednej strony były czytelne dla użytkowników z całego świata, a z drugiej dawały pełną informacje (z uwzględnieniem poprawnej pisowni) użytkownikom lokalnym. Niedoskonałość istniejących implementacji (korzystanie z ustalonych tablic kodowych) zmusza nas do stosowania rozwiązań tymczasowych.

3 Definiowanie obiektów bazy X.500

W niniejszym rozdziale korzystamy w szerokim zakresie ze specyfikacji standardu X.500 [CCITT]. Zajmujemy się jedynie modelem informacyjnym. Opis modelu funcjonalnego zawarty jest w naszym raporcie [GW], jak również w [CCITT] i [ROSE].

X.500 jest bazą „obiektową". Obiekty (entries) definiowane są w oparciu o pojęcie klasy. Określa się klasy podstawowe, a następnie ich podklasy. Niezbędnym atrybutem każdego obiektu jest `objectClass`, determinujący rodzinę, do której obiekt należy, a także konieczne i opcjonalne atrybuty opisujące ten obiekt.

Definicja obiektów bazuje na pojęciu dziedziczenia klas, co oznacza, że klasa obiektu może zostać zdefiniowana jako podklasa wcześniej określonej. Podklasy dziedziczą atrybuty swojej nadklasy wprowadzając przy tym własne rozszerzenia. Atrybut `objectClass` jest wielowartościowy. Bardzo często stosowane jest tworzenie nowej podklasy w postaci połączenia przekroju kilku klas wyjściowych. Atrybuty obiektów muszą zachowywać określone reguły syntaktyczne, które są przetwarzane zgodnie z metodami dla nich określonymi. Istotną cechą X.500 Directory jest jego rozszerzalność — na bazie kilku wspólnych typów obiektów i atrybutów mogą zostać zdefiniowane nowe klasy i atrybuty.

3.1 Pojęcia podstawowe

Nazewnictwo, którego używamy nie jest całkowicie standardowe i nie zawsze do końca jednoznaczne. Pojęcia *obiekt* używamy na ogół na oznaczenie obiektu świata fizycznego, który jest reprezentowany w bazie, termin ten jednak pojawia się również w innym znaczeniu, gdy mówimy o klasach i podobnych terminach związanych z „obiektowościa" bazy.

Termin *hasło* używany będzie na oznaczenie konkretnych pozycji w bazie danych, a więc reprezentacji obiektów. Hasło występuje również w X.500 jako atrybut obiektu (np. `userPassword`), nie będziemy jednak tutaj wspominać o tym zastosowaniu.

Poniżej przedstawiamy listę najważniejszych definicji pojęć X.500 Directory.

1. **Baza informacyjna** (Directory Information Base — DIB): całość informacji udostępniona przez X.500 (zbiór wszystkich obiektów — haseł).

2. **Drzewo informacji** (Directory Information Tree — DIT): Hierarchia, według której uporządkowana jest informacja zawarta w DIB.

3. **Hasło** (entry): węzeł DIT — fragment informacji zawartej w DIB opisujący pojedynczy obiekt.

4. **Hasło podstawowe** (object entry): hasło opisujące dany dany obiekt. Przyporządkowanie

   $$\texttt{obiekt} \mapsto \texttt{hasło podstawowe}$$

   stanowi bijekcję (odwzorowanie wzajemnie jednoznaczne) BIB na zbiór haseł podstawowych.

5. **Alias**: hasło będące odsyłaczem do hasła podstawowego. Występuje w DIT w celu nadania obiektowi innej nazwy niż ta, związana z hasłem podstawowym. (Standard 93 dopuszcza aliasy odsyłające do kolejnego aliasu, pod warunkiem ostatecznej redukcji do hasła podstawowego).

6. **Hasło bezpośrednio nadrzędne** (immediately superior entry): hasło umiejscowione w DIT bezpośrednio ponad hasłem rozważanym.

7. **Klasa obiektu** (object class): rodzina, do której należą obiekty o wspólnych cechach.

8. **Atrybut** (attribute): element informacji dotyczący obiektu i występujący jako fragment opisu hasła.

9. **Typ atrybutu** (attribute type): składowa atrybutu wskazująca na rodzaj informacji przedstawianej poprzez dany atrybut.

10. **Wartościowanie atrybutu** (attribute value assertion — AVA): przypisanie typowi atrybutu wartości.

11. **Wyróżniona wartość atrybutu** (distiguished value): jedna z wartości przypisanych atrybutowi, decydująca o unikalności hasła na danym poziomie poddrzewa i tworząca część jego nazwy.

12. **Relatywna nazwa wyróżniona** (Relative Distinguished Name — RDN): zespół (wyróżnionych) wartości atrybutów tworzących jednoznaczną nazwę na danym poziomie podgałęzi drzewa. Typowo stosuje się RDN skladające się z pojedynczych wartości atrybutów, ale dopuszczalne jest kojarzenie kilku różnych atrybutów i ich wartości.

13. **Nazwa wyróżniona** (Distinguished Name — DN): ciąg RDN określający umiejscowienie hasła w hierarchii DIT. Ciąg pusty oznacza korzeń drzewa.

3.2 Model informacyjny bazy X.500

X.500 Directory, podobnie jak inne aplikacje OSI, stosuje specyfikację CCITT — ASN.1 ([ASN.1]) do opisu struktur danych, tj. klas obiektów, atrybutów i zbiorów atrybutów.
ASN.1 daje możliwość definiowania makr, mechanizm ten pozwala na rozszerzenie zbioru określeń stosowanych do opisu złożonych danych.

Poniżej przedstawiamy ogólną postać definicji podstawowej infrastruktury informacyjnej X.500 za pomocą ASN.1.
Klasa obiektów została określona przez makro `OBJECT-CLASS`, w którym uwzględniono:

1. opcjonalne — podanie identyfikatora obiektu danej klasy;
2. wskazanie wszystkich podklas definiowanej klasy (`SUBCLASS OF`);
3. podanie listy obowiązkowych typów atrybutów w danej klasie (`MUST CONTAIN`) oraz typów atrybutów opcjonalnych (`MAY CONTAIN`).

Podstawowa klasa obiektów, będąca „nadklasą” każdej innej klasy to `top` o następującej definicji:

```
Top ::=
  OBJECT-CLASS
    MUST CONTAIN {ObjectClass}
```

Przykładowo definicja obiektu `alias` ma postać:

```
alias ::=
  OBJECT-CLASS
    SUBCLASS OF top
      MUST CONTAIN {AliasedObjectName}
```

Standard definiuje około 20 podstawowych klas obiektów.
Typy atrybutów są określane poprzez makro `ATTRIBUTE`, które:

1. przypisuje identyfikator obiektu typowi atrybutu;
2. wskazuje lub definiuje syntaktykę typu atrybutu (`WITH ATTRIBUTE-SYNTAX`);
3. wskazuje czy atrybut tego typu jest jednowartościowy (`SINGLE VALUE`), czy może byc wielowartościowy (`MULTI VALUE`).

Atrybut `ObjectClass` jest opisywany w notacji ASN.1 jako:

```
ObjectClass ::=
  ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX objectIdentifierSyntax
```

Brak określenia `SINGLE VALUE` implikuje wielowartościowość atrybutu.
Standard definiuje około 40 podstawowych typów atrybutów.
Sama składnia atrybutu — `ATTRIBUTE-SYNTAX` jest także zdefiniowana przez odpowiednie makro, w ramach którego następuje:

1. przypisanie identyfikatora syntaksowi atrybutu;
2. wskazanie typu danych poprzez isniejący już w ASN.1 syntaks;
3. opcjonalnie — definicja reguł dotyczących dopasowywania wartości atrybutów (`MATCHES FOR`); dopuszcza się tu trzy możliwości: równość (`EQUALITY`), podłańcuch (`SUBSTRINGS`), uporządkowanie (`ORDERING`). Nowa wersja standardu 93 znacznie rozbudowuje definicję reguł dopasowywania.

Na przykład:

```
telephoneNumberSyntax
  ATTRIBUTE-SYNTAX
    PrintableString
      SIZE(1..ub-telephone-number))
    MATCHES FOR EQUALITY SUBSTRINGS
::= {attributeSyntax 12}
```

Klauzula `SIZE` pozwala wyspecyfikować ograniczenia dotyczące wartości atrybutu, w tym przypadku została określona dopuszczalna długość łańcucha (ograniczenia te są podane w aneksie do rekomendacji X.520: „The Directory — selected attribute types" jako *UpperBounds*).

Standard X.500 wprowadza również makro `ATTRIBUTE-SET` umożliwiające grupowanie typów atrybutów. Makro zawiera klauzulę `CONTAINS`, po której wymieniane są typy atrybutów należące do grupy. Podstawowa definicja składa się z 4 zbiorów atrybutów, m.in. grupa `postalAttributeSet` gromadzi atrybuty potrzebne do opisania informacji o adresie obiektu.

```
postalAttributeSet
  ATTRIBUTE-SET
    CONTAINS {
      postalDeliveryOfficeName,
      postalAddress,
      postalCode,
```

```
      postalOfficeBox,
      streetAddress}
::= {attributeSet 1}
```

Standard X.500 93 pozwala na definiowanie zbiorów atrybutów w ramach makra `ATTRIBUTE`.

Formalnie, według notacji ASN.1 atrybut hasła ma postać:

```
AttributeValue ::= ANY

AttributeType ::= OBJECT IDENTIFIER

Attribute ::=
   SEQUENCE {
       type       AttributeType,
       values     SET OF AttributeValue}

RelativeDistinguishedName ::=
    SET OF SEQUENCE {
        AttributeType, AttributeValue}

RDNSequence ::=  SEQUENCE OF RelativeDistinguishedName

DistinguishedName ::= RDNSequence
```

Elementy bazy X.500, takie jak klasa obiektu czy atrybut, definiowane są w ASN.1 za pomocą typu zwanego identyfikatorem obiektu (`OBJECT IDENTIFIER`), który określa autorytatywnie nazwany obiekt. Identyfikatory te uporządkowane są również w strukturę drzewa, mają postać ciągu nieujemnych liczb całkowitych, z których każda jest etykietą kolejnego węzła drzewa na drodze od korzenia do bieżacej pozycji. Postaci numerycznej opisującej węzeł towarzyszy na ogół krótki opis tekstowy.

Podstawowy węzeł drzewa identyfikatorów obiektów (root) ma trzy węzły podporządkowane:

- `ccitt(0)`, administrowany przez CCITT;
- `iso(1)`, administrowany przez ISO/IEC;

- `joint-iso-ccitt(2)`, administrowany wspólnie przez ISO/IEC i CCITT.

co ASN.1 opisuje następująco:

```
ccitt            OBJECT IDENTIFIER ::= { 0 }
iso              OBJECT IDENTIFIER ::= { 1 }
joint-iso-ccitt  OBJECT IDENTIFIER ::= { 2 }
```

Dalej np. na gałęzi `ccitt` zdefiniowano:

```
data             OBJECT IDENTIFIER ::= { ccitt 9 }
pss              OBJECT IDENTIFIER ::= { data  2342}
quipu            OBJECT IDENTIFIER ::= { pss 19200300 }
```

System taki pozwala na jednoznaczne przyporządkowanie identyfikatorów i delegowanie odpowiedzialności za administrowanie podgałęziami drzewa identyfikatorów w ramach organizacji krajowych, a dalej przez administratorów na poziomie instytucji itp.

Po przydzieleniu gałęzi drzewa, instytucja lub kraj, ma pełną swobodę w definiowaniu swoich własnych elementów. Oczywiście lokalnie dodane klasy (czy atrybuty) nie będą interpetowane przez użytkowników nie mających dostępu do właściwych definicji (dzięki jednoznaczności nie zachodzi jednak niebezpieczeństwo interpretacji fałszywej). W standardzie X.500 (88) podstawowe identyfikatory rozpowszechniane są za pomocą statycznych tablic, w wersji 93 przewiduje się umieszczenie definicji w samym drzewie informacji. Takie rozwiązanie spowoduje, że niezdefiniowane obiekty nie będą się w ogóle pojawiały.

Poniżej przedstawiamy sposób realizacji określenia podstawowego obiektu bazy, jakim jest osoba, przyjęty w konkretnej implementacji X.500 Directory — QUIPU. System ten składa się z trzech statycznych tablic definiujących drzewo identyfikatorów.

- Tabela **oidtable.gen** zawiera podstawowe definicje.

```
ccitt:                 0
joint:                 2
data:                  ccitt.9
ds:                    joint.5
pss:                   data.2342
```

```
ucl:                     pss.19200300
attributeType:           ds.4
standardObjectClass:     ds.6
pilot:                   ucl.100
pilotAttributeType:      pilot.1
```

- Tabela **oidtable.at** zawiera defincje atrybutów według schematu:

  `nazwa[,skrót]:  objectIdentifier :  syntaks`

  Syntaksy (reguły składniowe i związane z nimi metody translacji) zakodowane są bezpośredno w źródło programu, stanowią część biblotek ISODE.

```
commonName,cn:           attributeType.3:  CaseIgnoreString
surname,sn:              attributeType.4:  CaseIgnoreString
description:             attributeType.13: CaseIgnoreString
telephoneNumber:         attributeType.20: TelephoneNumber
seeAlso:                 attributeType.34: DN
userPassword:            attributeType.35: Password
```

- Tabela **oidtable.oc** definjuje klasy (`objectClass`) według schematu:

  `nazwa:objectIdentifier:atrybuty konieczne:atrybuty dopuszczalne`

```
top:         standardObjectClass.0 : : objectClass :

person:      standardObjectClass.6 : top : CN ,surname : \
             description, seeAlso, telephoneNumber, userPassword
```

W implementacji QUIPU syntaksy są definiowane poprzez odpowiednie struktury danych i zawierają oprócz nazwy (pochodzącej z tablicy *oidtable.at*) adresy procedur wykonujących podstawowe operacje na tych atrybutach, tj. kodowanie, dekodowanie, alokacje, dealokacje, wypisanie, porównanie. Istnieje również możliwość wyspecyfikowania tu procedury odpowiedzialnej za realizację przybliżonego dopasowywania. Inicjalizacja opisanych struktur następuje poprzez wywołanie funkcji `add_attribute_syntax`, której zasada działania ma postać typową dla techniki programowania obiektowego (definiuje dane i metody).

4 Standard T.61

Jedną z reguł syntaktycznych stosowanych w X.500 jest wykorzystanie standardu telekomunikacyjnego T.61 do kodowania wszelkich znaków diakrytycznych. Łańcuchy zawierające wyłącznie znaki czytelne mają przypisany syntaks `PrintableString`, te w których dopuszcza się znaki specjalne określa się jako `T61String`. Tego typu łańcuch charakteryzuje się nagłówkiem w postaci identyfikatora {T.61} i zawiera, poza standardowymi znakami ASCII kody znaków diakrytycznych.

Narodowe znaki specjalne z akcentami mają w standardzie T.61 postać dwóch bajtów: pierwszy to kod akcentu, drugi kod znaku podstawowego, inne są reprezentowane jako jeden bajt i występują w tabeli na pozycjach powyżej 127. Bardzo istotne jest, że T.61 jednoznacznie koduje znaki bez wykorzystania pojęcia strony tabeli kodowej. O znaczeniu tego faktu wspomnimy poniżej.

Oto fragment tabeli kodów.

Kod (hex)	Opis	Znak
c1	grave accent	\`
c2	accute accent	´
c3	circumflex	ˆ
c4	tilde	˜
c5	macron	
c7	dot	˙
c8	diaresis	¨
c9	umlaut	¨
ca	ring	°
cb	cedilla	
cc	underline	_
cd	umlaut	¨
ce	ogonek	
e8	L z kreską (Ł)	Ł
ea	OE	Œ
f8	l z kreską (ł)	ł

5 Przybliżone dopasowywanie wzorców

Baza X.500 zawiera pojęcie dopasowania przybliżonego (approximate match). Jak już wspomniano, definicje zasad syntaktycznych określają również metody dopasowywania przybliżonego.

Dla atrybutów typu łańcuchowego stosuje się algorytm SOUNDEX. W założeniu algorytm ten powinien wyszukiwać wzorce podobnie brzmiące. Jego zasada działania opiera się na mechanizmie grupowania sekwencji znaków — słów podobnie brzmiących w klasy. Każda klasa otrzymuje swój kod, zaczynający się od pierwszego znaku słów, które reprezentuje. W celu dopasowania wielu słów muszą one wystąpić w porównywanym łańcuchu we właściwej kolejności.

Powyższa definicja nie jest uniwersalna i wymaga dopasowania do konkretnych języków. Konieczność stosowania różnych algorytmów nie jest przewidywana przez standard, tym niemniej powinna być zrealizowana jako warunek sprawnego funcjonowania bazy w lokalnych warunkach. Implementacja QUIPU pozwala w prosty sposób wprowadzić niezbędne rozszerzania w postaci dodatkowych procedur, do których odwoływać się będą metody obsługi syntaksów.

6 Ograniczenia płynące ze stosowania zestawów ośmiobitowych czcionek komputerowych.

Wiekszość aktualnie funkcjonujących aplikacji komputerowych korzysta z pojęcia strony kodowej. W ten sposób użytkownik ustala, z jakiego zestawu czcionek będzie korzystał. Zmiana strony kodowej powoduje wyświatlanie innych symboli na wysokich miejscach (powyzej 127) tabeli kodów. Bezpośrednim efektem takiego podejścia jest niemożność wyświetlania na jednym ekranie jednocześnie znaków z różnych języków. Stosowanie tabeli kodowych jest szczególnie uciażliwe w przypadkach niejednoznaczych standardów (np. tabela Latin–2 (852) stosowana przez Microsoft i tabela ISO8859-2)

Zastosowanie w X.500 standardu T.61 pozwala na jednoznaczne zakodowanie znaków łacińskich wszystkich języków europejskich, problemem pozostaje jednak ich wyświetlanie. Oczywiście nic nie stoi na przeszkodzie, aby aplikacja stosowała różne czcionki w ramach jednego ekranu i w ten sposób była całkowicie uniwersalna, ale nie jest to jak na razie powszechne podejście. Głównym powodem takiego zaniedbania jest najprawdopodobniej fakt, ze większość zachodnioeuropejskich języków mieści sie w ramach podstawowej tableli ISO8859-1.

7 Proponowany schemat dostosowania bazy X.500 do współpracy z językami narodowymi na przykładzie języka polskiego.

7.1 Polskie modyfikacje a międzynarodowy dostęp

Zakładamy, że w obecnej chwili typowy użytkownik na świecie ma do dyspozycji interfejs wyświetlający znaki zgodnie z tabelą ISO-Latin-1. Oznacza to, że nie ma dostępu do polskich znaków diakrytycznych i jego interfejs będzie je zastępował znakami nieczytelnymi. Uważamy zatem, że tymczasowo użytkownik spoza Polski musi bazować na polskich nazwach poddanych obraniu ze znaków diakrytycznych i takie właśnie nazwy mają być stosowane w podstawowych atrybutach bazy X.500 Directory.

Jednocześnie jednak poprawna polska pisownia musi być w Directory przechowywana, zarówno ze względu na przyszłość, kiedy być może będzie poprawnie interpretowana, jak dla wykorzystania lokalnego w Polsce. Realizację dostępu do właściwej pisowni powinny oferować specjalnie do tego dostosowane interfejsy użytkowe. Dla wprowadzania polskich danych do bazy X.500 definiujemy specjalne atrybuty duplikujące w zasadzie atrybuty podstawowe, ale zawierające poprawne nazwy. Atrybuty będące lokalnymi rozszerzeniami nie będą poza polską widziane (z powodu niemożliwości zdekodowania identyfikatrów je określających), specjalizowane polskie DUA powinny natomiast korzystąc z tych atrybutów w pierwszej kolejności, ignorując wartości atrybutów „obranych”.

7.2 Polskie rozszerzenia

Poniżej przedstawiamy proponowane rozszerzenia, związane z definicją nowych klas obiektów i atrybutów w celu wprowadzania nazw polskich.

- Podstawowe polskie identyfikatory (`oidtable.gen`).

```
nask:                    data.2602
umk:                     nask.1
polishQuipu:             umk.1
polishObjectClass:       polishQuipu.1
polishAttributeType:     polishQuipu.2
```

- Atrybuty (`oidtable.at`).

```
polishCnName,plcn:       polishAttributeType.1 :caseIgnoreString
polishOName,plo:         polishAttributeType.2 :caseIgnoreString
polishOuName,plou:       polishAttributeType.3 :caseIgnoreString
polishLName,pll:         polishAttributeType.4 :caseIgnoreString
polishDescription:       polishAttributeType.5 :caseIgnoreList
polishInfo:              polishAttributeType.6 :caseIgnoreList
polishPostalAddress:     polishAttributeType.7 :PostalAddress
polishTitle:             polishAttributeType.8 :caseIgnoreString
multiLineDescription:    polishAttributeType.11:caseIgnoreList
fullOName,fullO:         polishAttributeType.12:caseIgnoreList
fullPolishOName,fullpLo: polishAttributeType.13:caseIgnoreList
fullOuName,fullOu:       polishAttributeType.14:caseIgnoreList
fullPolishOuName,fullpLo:polishAttributeType.15:caseIgnoreList
```

- Klasy (`oidtable.oc`).

```
polishPerson: polishObjectClass.1 : person :\
   polishCnName :\
   polishDescription, polishPostalAddress, polishInfo,\
   polishTitle, multiLineDescription, memberOfGroup

polishOrganization: polishObjectClass.2 : organization :\
   polishOName :\
   polishDescription, polishPostalAddress,polishInfo,\
   polishLName, fullOName, fullPolishOName

polishOrganizationalUnit: polishObjectClass.3 :\
   organizationalUnit :\
   polishOuName : polishDescription, polishPostalAddress,\
   polishInfo, polishLName, fullOuName, fullPolishOuName

polishLocality: polishObjectClass.4 : locality :\
   polishLName :\
   polishDescription, polishPostalAddress, polishInfo
```

Należy zwrócić uwagę na potrzebę stosowania syntaksu `caseIgnoreList` do definicji pełnych nazw w języku polskim (`fullPolishOName, fullCnName` itp.). Wynika to z narzuconego przez standard ograniczenia dla `caseIgnoreString`, który może być łańcuchem o maksymalnej długości 64 znaków. Nietrudno, stosując kodowanie według T.61 tę granicę przekroczyć, stąd konieczność definiowania jako `caseIgnoreList`, pozwalającego wprowadzać łańcuch złożony ze składowych, z których każda ma do 64 znaków.

7.3 Zasady porównywania atrybutów

Jedną z najistotniejszych funkcji wykonywanych na zbiorach danych X.500 jest przeszukiwanie. Polskie serwery (DSA) powinny wykorzystywać zmodyfikowany wariant algorytmu przybliżonego dopasowywania wzorców biórący pod uwagę specyfikę polskiego języka. W chwili obecnej algorytm tem musi być zakodowany w sewerze jako `MatchingRule` dla odpowiedniego typu atrybutu (a dokładniej syntaksu).

7.4 Dostosowane interfejsy użytkownika

Polski użytkownik powinien być wyposażony w zmodyfikowane interfejsy DUA poprawnie wyświetlające znaki z zakresu tabeli ISO-Latin-2 i przetwarzające znaki spoza tej tabeli w sposób nie zaburzający wydruku (i nie wprowadzający mylnej informacji).

W ramach oprogramowania QUIPU translacja odbywa się w modułach bibliotecznych, a zatem wystarczy rozpowszechniać poprawione biblioteki. Część prac związanych z implementacją tych funkcji w odpowiednich modułach została już wykonana.

Innego podejścia wymagają interfejsy korzystające z uproszczonej wersji protokołu X.500, zwanego LDAP (Lightweight Directory Access Protocol). Protokół ten został zdefiniowany jako komplementarny w stosunku do pełnego DAP, ale pozbawiony zbędnego balastu i nastawiony na prostą interakcję. Powstało wiele interfejsów opartych na tym protokole, jako że tworzenie ich jest mniej skomplikowane. Niestety funkcje interpretacji łańcuchów T.61 nie są wbudowane w biblioteki LDAP. W tym przypadku konieczne jest modyfikowanie każedgo programu użytkowego oddzielnie.

Przewiewidujemy, że w niedługim czasie, w ramach prac objętych grantem KBN udostępnimy kilka zmodyfikowanych interfejsów. W drugiej kolejności pojawią się interfejsy ukierunkowane na polskie atrybuty w zakresie przeszukiwania bazy.

Bibliografia

[X500] Data Communication Networks: Directory, Recommendations X500-X.521, CCITT, Fascile VIII.8 of Blue Book.

[ASN1] D. Steedman, ASN.1 The Tutorial and Reference, 1990.

[GW] Maria Górecka, Tomasz M. Wolniewicz, Efektywność rozproszonych baz danych na podstawie implementacji X.500 Directory, 1994.

[ROSE] Marshall T. Rose, The Little Black Book: Mail Bounding with OSI Directory Services, Prentice Hall, 1992.