ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: INFORMATYKA z. 35 Nr kol. 1402

Zygmunt BOK
Zakłady Tworzyw Sztucznych 'NITRON' S.A.

INTEGRACJA PROCESÓW ZARZĄDZANIA W ZASTANYCH PRZEMYSŁOWYCH SYSTEMACH INFORMATYCZNYCH

Streszczenie. W tym artykule - na przykładzie przemysłowego systemu informatycznego pracującego w Zakładach Tworzyw Sztucznych „Nitron" S.A. pod kontrolą sieciowego systemu operacyjnego Novell NetWare 4.x - przedstawiono metodę dostępu do informacyjnych baz danych, opartą na obiektowo-zorientowanej bazie NDS oraz technologii ODBC. Bazując na tej metodzie zintegrowano proces zarządzania istniejącymi zasobami sieciowymi typu użytkownicy, serwery plików, drukarki itd. z procesem zarządzania przemysłowymi systemami informacyjnymi współpracującymi z bazami danych typu *dBaseIII*, *Btrieve* zainstalowanych w systemach plików serwerów sieciowych - w jednolite pod względem zarządzania środowisko sieciowe.

INTEGRATION OF MANAGEMENT PROCESSES IN INDUSTRIAL LEGACY INFORMATION SYSTEMS

Summary. In this article - on example industrial information system working in ZTS 'Nitron' S.A. factory under network Novell NetWare 4.x operating system control, an access method to information databases based on object-oriented database NDS and ODBC technology was presented. Based on this method a resource management process such as users, file servers, printers etc. together with industrial information management systems that controls *dBaseIII* and *Btrieve* type databases installed in file systems on network file servers has been integrated into uniform under management network environment.

1. Wprowadzenie

Wraz z wprowadzeniem przez firmę Novell na rynek w marcu 1993 r. następnej generacji sieciowego systemu operacyjnego NetWare 4.0, a wraz z nim tzw. - Usług Katalogowych [1]

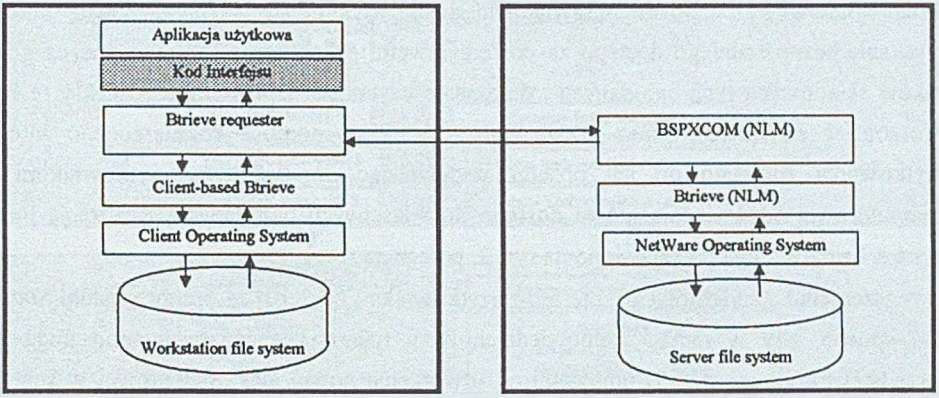
(*Novell Directory Services* - NDS), pojawiła się możliwość zintegrowania za pomocą NDS'u różnych zasobów sieciowych w jednolite i zarazem proste w zarządzaniu oraz wykorzystaniu środowisko sieciowe. Usługi Katalogowe (*Directory Services*) stanowią pewną bazę informacyjną używającą zróżnicowanych typów informacji o użytkownikach oraz zasobach w sieciowym środowisku obliczeniowym, natomiast NDS jest obiektowo zorientowaną implementacją tej bazy [2], przechowującą informacje o wszystkich obiektach znajdujących się w sieci. Niniejszy artykuł - na przykładzie przemysłowego systemu informatycznego pracującego pod kontrolą sieciowego systemu operacyjnego Novell NetWare 4.x w Zakładach Tworzyw Sztucznych „Nitron" S.A., przedstawia metodę dostępu do informacyjnych baz danych, wykorzystującą obiektowo zorientowaną bazę NDS oraz technologię ODBC, za pomocą której można zintegrować proces zarządzania istniejącymi zasobami sieciowymi typu użytkownicy, serwery plików, serwery drukowania, kolejki, drukarki itd. z procesem zarządzania przemysłowymi systemami informacyjnymi opartymi na bazach danych typu *dBaseIII, Btrieve* zainstalowanych w systemach plików serwerów sieciowych - w jednolite pod względem zarządzania środowisko sieciowe.

Systemy informacyjne oparte na bazach danych tych typów, zainstalowane w sieci Novell NetWare 4.x, są jeszcze w chwili obecnej z różnych względów dość mocno rozpowszechnione i nadal eksploatowane w wielu małych i średnich zakładach przemysłowych. Przejście w tych zakładach na nowoczesne systemy klasy MRP II do obsługi procesów finansowo-gospodarczych w skali całego przedsiębiorstwa jest procesem długotrwałym, wymagającym determinacji kierownictwa oraz zespołu wdrażającego, znacznych nakładów sił i środków - często przekraczających możliwości finansowe i kadrowe przedsiębiorstwa. Z tego też powodu służby informatyczne w wielu spośród tych zakładów sprawują nadzór nad zastanymi i w dalszym ciągu eksploatowanymi systemami informacyjnymi, tj. aplikacjami użytkowymi oraz plikami baz danych przez nie obsługiwanych, często bez stosownych narzędzi do ich obsługi - w szczególności w przypadku plików typu *Btrieve*.

W sytuacjach awaryjnych, kiedy została naruszona spójność bazy danych (nadmiar lub brak rekordów w pewnych plikach) - szczególnie dotyczy to baz danych typu *dBaseIII* obsługiwanych przez aplikacje użytkowe, w których nie zastosowano mechanizmu transakcji, powstaje problem, w jaki sposób usunąć niespójność bazy danych. Do naruszenia spójności bazy danych dochodzi najczęściej, kiedy aplikacja użytkowa próbuje dodać nowy lub skasować istniejący rekord z pewnego pliku .DBF, w sytuacji kiedy powinien on być lub aktualnie jest powiązany z innymi rekordami w innych plikach. Do naruszenia spójności bazy dochodzi również podczas jednoczesnej modyfikacji danych pochodzących z rekordów kilku plików połączonych za pomocą klucza poleceniem SET RELATION TO. W przypadkach skrajnych może nie dojść do odpowiednich zmian w rekordach znajdujących się plikach

wymienionych w poleceniu SET, co natychmiast objawia się brakiem spójności bazy. O ile w przypadku plików typu *dBaseIII*, w celu usunięcia niespójności bazy danych możliwe jest użycie prostego interpretatora poleceń - np. programu *dbase.exe* f-my Ashton-Tate- (jeśli istnieje w firmie biorąc pod uwagę różne zaszłości), to w przypadku plików Btrieve'owskich jedyną alternatywą jest użycie specjalistycznego narzędzia programistycznego - Xtrieve Plus f-my Novell współpracującego z systemem obsługi zbiorów dyskowych (*Novell's Btrieve Record Manager*) [3,4] opartym na metodzie indeksowania tzw. B-drzew, z wbudowanymi mechanizmami ochrony na błędne działanie sprzętu.

Architektura logiczna tego systemu w wersji klient-serwer, którą przedstawiono na rys. 1, składa się z komponentów pozwalających aplikacji użytkowej na dostęp do danych zawartych w systemie plików zainstalowanym lokalnie na stacji roboczej lub na serwerze sieciowym.



Rys. 1. Architektura logiczna systemu obsługi plików dyskowych typu *Btrieve*
Fig. 1. Logical architecture of the Btrieve disk file system management

Jak widać z tego rysunku, oprócz elementów stałych istnieje również element zmienny, który jest różny dla różnych języków programowania - tzw. kod interfejsu. Jest to kod złączony razem z aplikacją użytkową podczas procesu kompilacji i konsolidacji wykonujący operacje I/O na plikach bazy danych za pomocą odpowiednich funkcji Btrieve'owskich, po wykonaniu których - w odpowiedzi - operacje te zwracają odpowiednią wartość w zmiennej statusowej typu *integer* lub innymi słowy, zwracają tzw. kod statusu [5] wykonanej operacji. Po wywołaniu określonej funkcji Btrieve'a aplikacja powinna zawsze sprawdzić wartość kodu statusu. Jeśli wartość kodu statusu równa się zero, to operacja na pliku zakończyła się sukcesem, natomiast wszystkie niezerowe wartości kodu statusu wskazują na pewne błędy i aplikacja powinna sama je rozpoznać oraz odpowiednio je obsłużyć.

Biorąc powyższe pod uwagę, jeśli służby informatyczne nie zostały wyposażone przez dostawcę oprogramowania użytkowego w specjalny program do indywidualnej obsługi plików

Btrieve'owskich wchodzących w skład bazy danych, to pozostaje im jedno wyjście - prośba o pomoc programistów dostawcy oprogramowania.

2. Opis metody

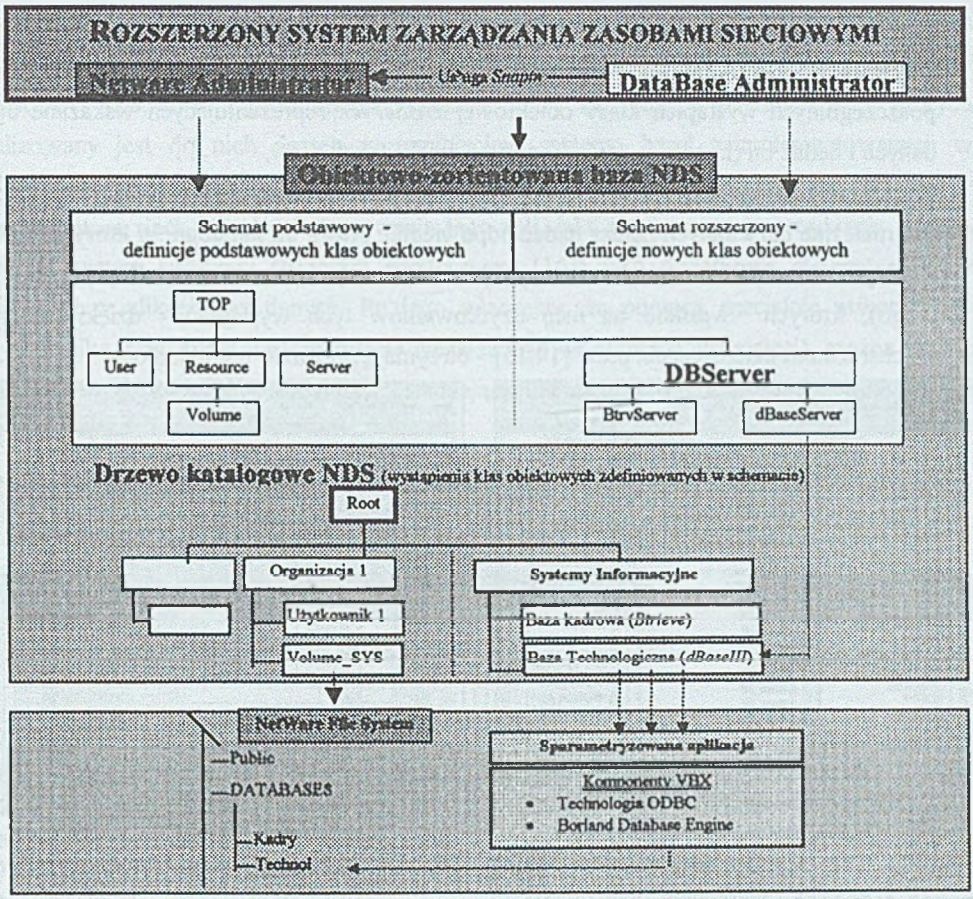
Proponowana metoda opiera się na założeniu, że jeśli każdy zasób sieciowy w sieci Novell NetWare 4.x reprezentowany jest przez pewien obiekt NDS'u tworzony i zarządzany poprzez graficzny interfejs użytkownika znany pod nazwą *NetWare Administrator*, to można również utworzyć inne obiekty w katalogowej bazie NDS [6], które przyporządkowane będą pewnym informacyjnym bazom danych zainstalowanym na pewnych serwerach sieciowych o takich właściwościach, które uprawnionym użytkownikom sieci komputerowej umożliwią uzyskanie bezpośredniego dostępu za pośrednictwem technologii ODBC do poszczególnych plików składowych tych baz danych i dokonanie czynności naprawczych. Obiekty te byłyby tworzone i zarządzane przez administratora sieci za pomocą rozszerzonego interfejsu użytkownika zdolnego do ich obsługi, zapewniając uprawnionym użytkownikom sieci komputerowej odpowiednie prawa dostępu do właściwych baz danych. Realizacja metody, którą schematycznie przedstawiono na rys. 2, polega na:

1) rozszerzeniu standardowego interfejsu użytkownika [7,8,9,10] za pomocą usługi *Snapin* w taki sposób, aby w ramach odpowiednich opcji menu uruchamianych spod opcji *Tools* (przedstawionych na rys. 3) umożliwił on utworzenie nowej klasy obiektowej w schemacie NDS [6,10,11],

2) rozszerzeniu schematu podstawowego NDS o nową klasę obiektową typu *DBServer* przedstawioną na rys. 4, wyprowadzoną z klasy obiektowej *Server*, i utworzenie jej wystąpień (instancji) [12,21] reprezentujących wskazane bazy danych (co przedstawiono na rys. 5) zainstalowane w rzeczywistym drzewie katalogowym NDS,

3) wykonaniu sparametryzowanej obiektowej aplikacji uruchamianej z poziomu podprogramu obsługującego te wystąpienia, która zapewni dostęp do danych we wskazanych plikach (przedstawionych na rys. 6) pewnej bazy danych. Dostęp do rekordów w tych plikach (przedstawiony na rys.7) realizowany jest za pomocą specjalizowanych komponentów typu *TVbxControl* [13], znajdujących się w bibliotece *Object Windows Library* kompilatora Borland C++ v.5.02, w których zaimplementowano mechanizmy *Borland Database Engine* oraz technologii ODBC.

Jednym z bardzo ważnych problemów, które pojawiają się w przypadku wdrożenia proponowanej metody, jest ochrona informacji zawartych w plikach składowych baz danych reprezentowanych przez wystąpienia klasy *DBServer*.



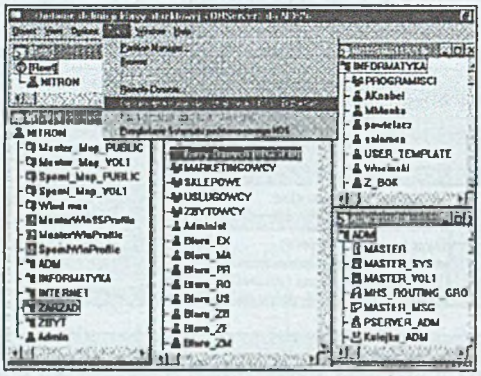
Rys. 2. Integracja procesów zarządzania zasobami sieciowymi i systemami informacyjnymi
Fig. 2. The integration of the network resources and information systems management process

Można go rozwiązać stosując jeden z następujących trzech wariantów wyboru użytkowników uprawnionych do dokonania czynności naprawczych:

- Spośród użytkowników sieciowych aktualnie mających dostęp (odpowiednie prawa) do katalogów systemu plików pewnego serwera sieciowego, w których zainstalowano pliki bazy danych pewnego systemu informacyjnego należy wybrać tych, którzy za pomocą opisywanej metody powinni otrzymać do niej dostęp w celu wykonania czynności naprawczych. Może to być również administrator tego systemu informacyjnego (jeśli istnieje) lub w ostateczności administrator całej sieci komputerowej. Ten wariant zastosowano w proponowanej metodzie i przedstawiono go na rys. 8.
- Spośród użytkowników sieciowych nie mających żadnych praw do eksploatacji wskazanych systemów informacyjnych (informatycy, programiści) wybrać tych, którzy

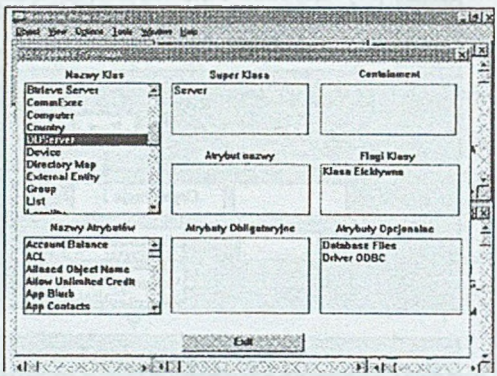
powinni zyskać niezbędne prawa do dokonania czynności naprawczych na plikach składowych bazy danych, po czym wpisać należy ich na listę użytkowników poszczególnych wystąpień klasy obiektowej *DBServer* reprezentujących wskazane bazy danych i nadać im (każdemu osobno) zestaw niezbędnych praw.

- Poszczególnym wystąpieniom klasy obiektowej *DBServer* reprezentującym pewne informacyjne bazy danych należy nadać odpowiednie prawa do katalogu, w którym te bazy zostały zainstalowane, wówczas wszyscy użytkownicy (nawet bez żadnych praw do niego), których wpisano na listę użytkowników tych wystąpień - dzięki ogólnym mechanizmom dziedziczenia praw [14,15] - otrzymują prawa do tego katalogu.



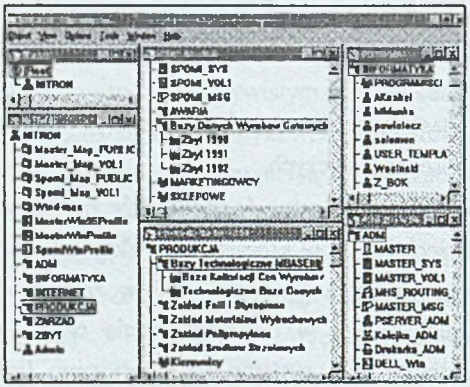
Rys. 3. Rozszerzony interfejs użytkownika, umożliwiający utworzenie nowej klasy obiektowej w schemacie NDS

Fig. 3. The extended user interface which enables create new object class in the NDS schema



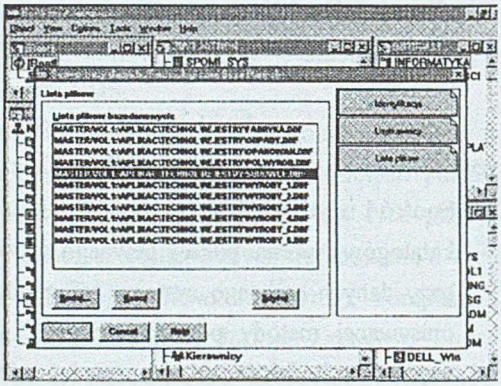
Rys. 4. Nowa klasa obiektowa *DBServer* wyprowadzona z klasy obiektowej typu *Server*

Fig. 4. New object class DBServer derived from *Server* object type class



Rys. 5. Bazy danych reprezentowane przez wystąpienia klasy obiektowej *DBServer*

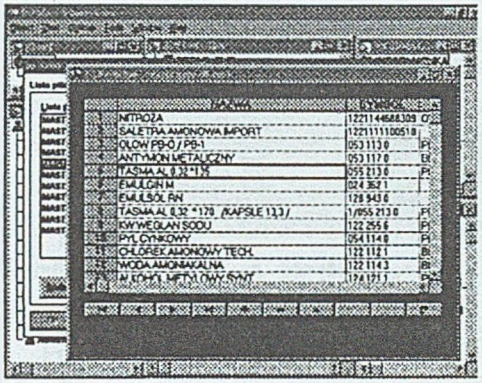
Fig. 5. Databases represented by *DBServer* class object instances



Rys. 6. Lista plików składowych wyróżnionej bazy danych

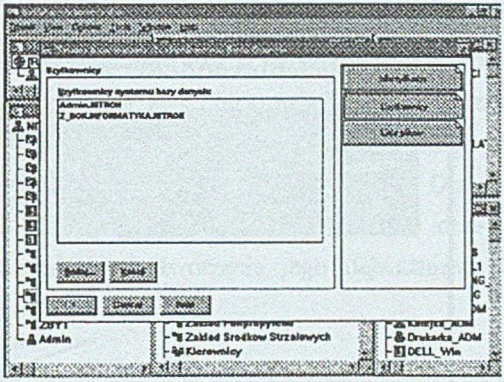
Fig. 6. The distinguished database component files list

Realizacja powyższych wariantów wymaga jednak pewnego zaufania w stosunku do osoby pełniącej funkcje administratora sieci, który tworzy i inicjalizuje nowe wystąpienia klasy obiektowej DBServer reprezentujące wskazane bazy danych, za pośrednictwem których realizowany jest do nich dostęp z pominięciem systemu haseł zaimplementowanych w aplikacjach użytkowych. W przypadku braku lub ograniczonego zaufania do osoby administratora sieci, w celu zbalansowania jego roli, jaką pełni przy jej zarządzaniu, można włączyć system auditingu (program auditcon.exe [16]) jako dodatkowy element ochrony informacji w plikach baz danych. Po jego włączeniu, za pomocą specjalnie utworzonego użytkownika, tzw. audytora (całkowicie niezależnego od administratora sieci), można śledzić działalność jego oraz użytkowników mających prawa do dokonywania czynności naprawczych.



Rys. 7. Dostęp do rekordów zawartych w wyróżnionym pliku składowym bazy danych za pomocą komponentów typu *TVbxControl*

Fig. 7. Access to records contained in the distinguished database file by *TVbxControl* type components



Rys. 8. Lista użytkowników mających dostęp do wyróżnionej bazy danych

Fig. 8. The users list which have access to distinguished database

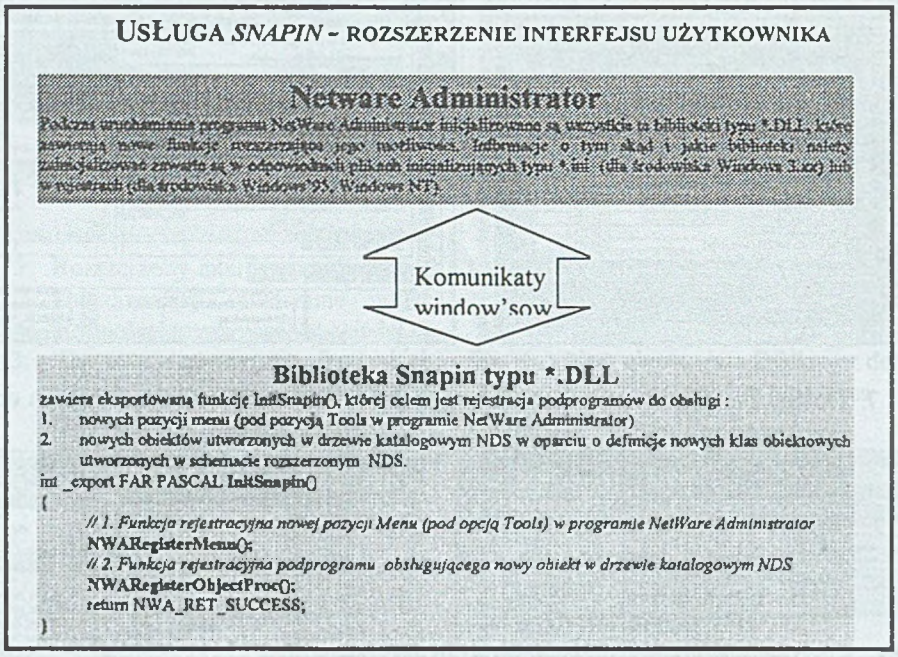
2.1. Rozszerzenie standardowego interfejsu użytkownika

Graficzny interfejs użytkownika, jako aplikacja działająca w środowisku Windows, będąca składnikiem pakietu systemu operacyjnego Novell NetWare 4.xx, może rozpoznać i obsłużyć klasy obiektowe oraz ich własności zdefiniowane w schemacie podstawowym dostarczanym wraz z NetWare 4.xx. W celu umożliwienia rozszerzenia [8] interfejsu użytkownika o nowe możliwości, tj.:

- dodanie nowej pozycji menu pod opcją *Tools*,
- dodanie nowego przycisku w pasku narzędziowym *Toolbar*,

- zapewnienie ikony reprezentującej nową klasę obiektową ze schematu rozszerzonego,
- zapewnienie nazw dla nowych klas obiektowych oraz ich własności,
- zdefiniowanie nowych okien dialogowych dla obsługi nowych klas obiektowych,
- zdefiniowanie dodatkowych nowych stron dialogowych dla istniejących klas obiektowych,

firma Novell opracowała usługę o nazwie *Snapin Services*. Usługa ta, pozwala istniejącym narzędziom administracyjnym, w tym standardowemu interfejsowi użytkownika, wykorzystywać rozszerzone schematy NDS, współpracujące z bibliotekami typu *.DLL zawierające kod dodatkowych funkcji rozszerzających jego możliwości. Biblioteki te, zwane także modułami typu *snapin*, ładowane są do pamięci operacyjnej oraz rejestrowane i inicjalizowane w trakcie jego uruchamiania. Ogólny zarys usługi *Snapin*, za pomocą której możliwe jest rozszerzenie programu *NetWare Administrator*, w poglądowy sposób przedstawiono na rys. 9.



Rys. 9. Zarys usługi *Snapin*
Fig. 9. The layout of the *Snapin* service

Aby wymiana informacji (dokonywana za pomocą standardowych komunikatów systemu operacyjnego windows, realizowanych za pomocą standardowej funkcji *SendMessage*) pomiędzy tym programem a bibliotekami typu *snapin* mogła zachodzić, biblioteki te muszą zawierać predefiniowane eksportowane funkcje *InitSnapin()*. Funkcje te używane są przez program *NetWare Administrator* do poinformowania bibliotek DLL, że jest on gotowy do zarejestrowania nowych rozszerzeń (podprogramów).

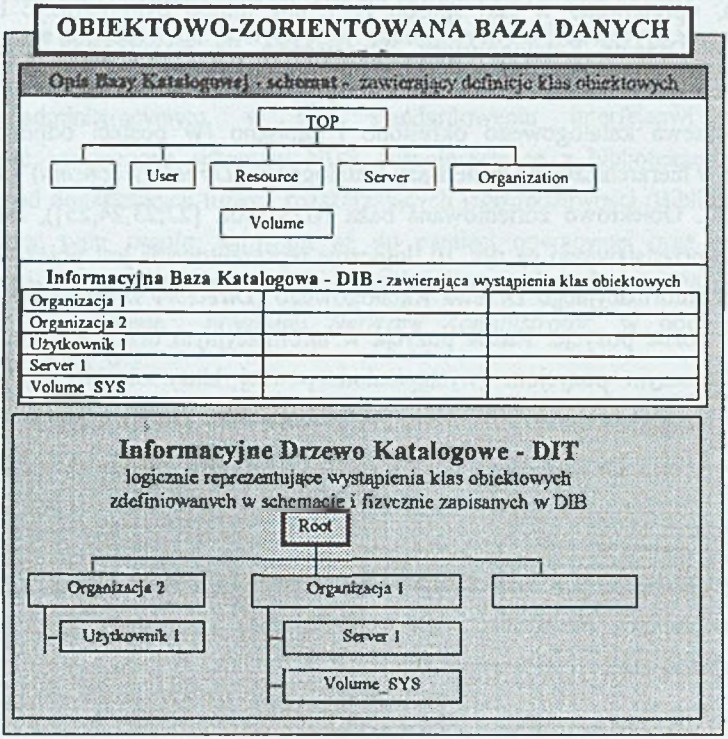
2.2. Rozszerzenie schematu NDS o nową klasę obiektową

NDS, będąca obiektowo zorientowaną implementacją bazy informacyjnej, tj. usług katalogowych, opracowano w taki sposób [17], aby można było tworzyć hierarchiczną strukturę tzw. Drzewa Katalogowego, składającego się z jednostek organizacyjnych zawierających użytkowników oraz inne zasoby sieciowe, natomiast zasady definiujące konstrukcję drzewa katalogowego określono i zapisano (w postaci odpowiednich klas obiektowych) w hierarchicznym Opisie Bazy Katalogowej (*Directory Schema*) lub inaczej - w schemacie [21]. Obiektowo zorientowana baza NDS (por. [22,23,24,25]), której budowę schematycznie przedstawiono na rys. 10, logicznie reprezentowana jest przez hierarchiczną strukturę [12] Informacyjnego Drzewa Katalogowego (*Directory Information Tree* - DIT) zawierającego różne pozycje. Każda pozycja w informacyjnym drzewie katalogowym DIT odpowiada fizycznym pozycjom (wystąpieniom pewnej klasy obiektowej) zawartym w Informacyjnej Bazie Katalogowej (*Directory Information Base - DIB*), która jest jego fizyczną reprezentacją. Można powiedzieć, że schemat jako zbiór klas obiektowych określa rodzaje obiektów, tj. wystąpienia tych klas, które można dodawać do Informacyjnej Bazy Katalogowej (*DIB*), rekordy której z kolei logicznie reprezentowane są przez Informacyjne Drzewo Katalogowego (*DIT*). Opis Bazy Katalogowej - schemat - kontroluje strukturę drzewa katalogowego poprzez określenie zasad kontrolujących tworzenie jego najważniejszych elementów składowych [11], tj.:

- klas obiektowych,
- typów atrybutów, wykorzystywanych do tworzenia konkretnych atrybutów,
- składni atrybutów.

Schemat NDS'u określa specyficzne typy danych oraz sposób, w jaki są one konstruowane oraz zapisywane w Informacyjnej Bazie Katalogowej (*DIB*), jaki obiekt (wystąpienie klasy obiektowej) NDS'u może istnieć, jakie atrybuty może zawierać, jakie wartości atrybutu są dozwolone oraz gdzie może istnieć w Informacyjnym Drzewie Katalogowym NDS'u. Informacyjne Drzewo Katalogowe tworzone jest z wystąpień klas obiektowych. Zbiór zasad kontrolujących tworzenie konkretnego typu obiektowego [18], zwany jest klasą obiektową, która według [19] jest opisem obiektu lub obiektów z jednolitym predefiniowanym zbiorem atrybutów i usług zawierający opis tworzenia nowych obiektów w klasie. Wszystkie atrybuty dowolnego obiektu oparte są na pewnym zbiorze wcześniej zdefiniowanych typów atrybutów [21], które z kolei definiowane są na podstawie predefiniowanych typów wartości, co schematycznie przedstawiono na rys. 11. Każda klasa obiektowa zdefiniowana w schemacie NDS określa rodzaje wystąpień, które mogą pojawić się w informacyjnym drzewie katalogowym. Każde wystąpienie (obiekt) w tym drzewie należy do pewnej klasy obiektowej

zdefiniowanej w schemacie, która określa jakie atrybuty i metody mogą być nim stowarzyszone.



Rys. 10. Budowa obiektowo-zorientowanej bazy NDS
Fig. 10. Construction of the object oriented database NDS

Innymi słowy, schemat ustanawia rodzaje wystąpień, które można dodawać do obiektowej bazy NDS. Tak więc wystąpienia oparte są na definicjach klas obiektowych, które składają się z typów atrybutów definiowanych za pomocą predefiniowanych typów wartości, tj. definicji syntaktycznych. Oprócz atrybutów, na kompletną definicję każdej klasy obiektowej składają się następujące elementy:

- elementy reguł strukturalnych, tj:
  - element *Nazwany przez*,
  - element *Zawieranie*,
- element *Nadklasy*,
- element *Atrybutów obligatoryjnych*,
- element *Atrybutów opcjonalnych*.

Elementy reguł strukturalnych - definiują dla poszczególnych wystąpień możliwe związki strukturalne pomiędzy nimi w Drzewie Katalogowym NDS'u. W szczególności dla każdej klasy obiektowej element:

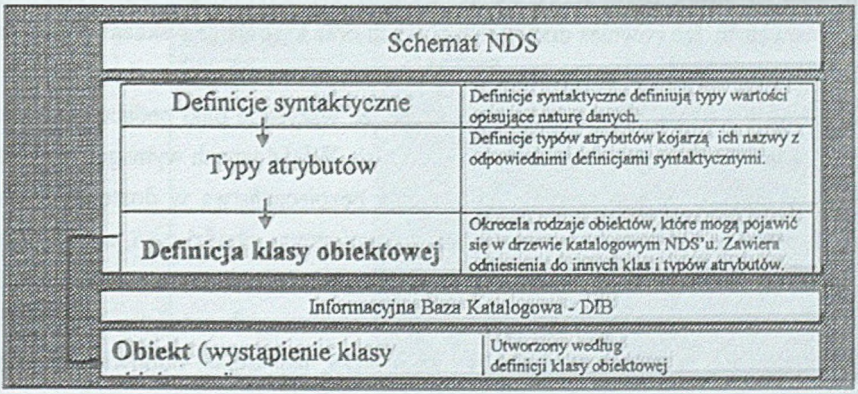
Nazwany przez (*Named By*) - definiuje komponent nazwy tego wystąpienia, który dla większości obiektów typu „liść" (*leaf-node*) przyjmuje wartość „CN" (*Common Name*),

Zawieranie (*Containment*) - definiuje położenie w stosunku do innych obiektów w drzewie katalogowym, tj. w jakim kontenerze może rezydować. Wykorzystywane są do zdefiniowania potencjalnych związków pomiędzy nimi,

Nadklasa (*Super Class*) - jest elementem służącym do określania struktury klas obiektowych w schemacie NDS'u. W szczególności element *Nadklasy* określa reguły dziedziczenia, tj. określa podstawową klasę obiektową, z której nowa klasa obiektowa jest wyprowadzana. Za jego pomocą następuje stowarzyszenie definicji klasy definiowanej z definicją jej nadklasy, w wyniku czego nowa klasa obiektowa dziedziczy wszystkie własności klasy podstawowej. Kompletna definicja każdej klasy obiektowej składa się z komponentów tej klasy oraz komponentów wszystkich nadklas znajdujących się (zapisanych explicite) na liście elementu *Nadklasy*. Klasy obiektowe znajdujące się na górze hierarchii zapewniają bardziej ogólną charakterystykę (własności), szczegółową natomiast zapewniają klasy obiektowe znajdujące się na jej dole.

Atrybuty obligatoryjne - definiują te atrybuty, których wartości muszą być określone przed utworzeniem obiektu (wystąpienia) danej klasy obiektowej w drzewie katalogowym NDS'u.

Atrybuty opcjonalne - definiują te atrybuty, których wartości nie muszą być określone przed utworzeniem obiektu danej klasy obiektowej. Ich określenie zwiększa użyteczność tego obiektu, pozwalając na zapisanie w nim więcej informacji.



Rys. 11. Wzajemne zależności pomiędzy obiektami, klasami obiektowymi, typami atrybutów oraz definicjami syntaktycznymi w schemacie NDS

Fig. 11. Mutually relationship between object class instances, object classes, attribute types and syntax definitions in the NDS schema

2.2.1. Projekt nowej klasy obiektowej DBServer do obsługi baz danych

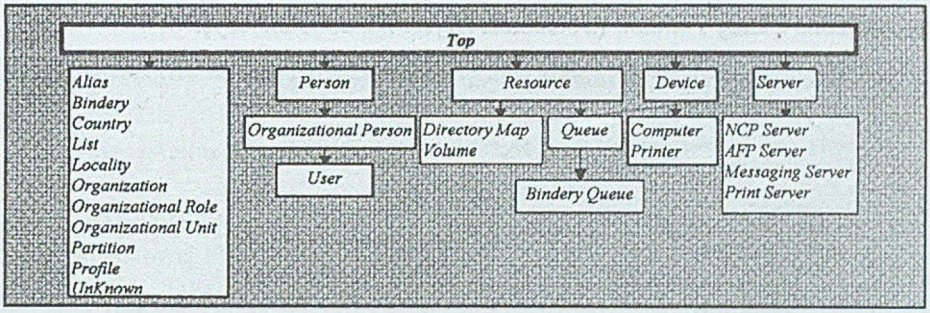
W celu realizacji zaproponowanej metody, niezbędnym elementem jest zaprojektowanie nowej klasy obiektowej na bazie klas obiektowych już istniejących w schemacie podstawowym NDS. Nowa klasa obiektowa [17] powinna w maksymalnym stopniu wykorzystywać własności odziedziczone z istniejących klas obiektowych i ich atrybutów. Wystąpienia tej klasy spełniałyby rolę logicznych reprezentantów baz danych użytkowych systemów informacyjnych, za pośrednictwem których możliwy byłby dostęp do ich tablic (plików). W ten sposób informacyjne bazy danych zainstalowane w systemach plików na pewnych serwerach sieciowych można by traktować jako nowe zasoby sieciowe (obok już istniejących typu użytkownicy, serwery plików, serwery druku itd.) zarządzane w jednolity sposób za pomocą rozszerzonego interfejsu użytkownika, tj. programu *NetWare Administrator* współpracującego z odpowiednimi bibliotekami typu *snapin*. Biorąc pod uwagę fakt, że nowa klasa, której wystąpienia mają współpracować ze specjalizowanymi komponentami typu *TVbxControl*, znajdującymi się w bibliotece *Object Windows Library*, w których zaimplementowano mechanizmy *Borland Database Engine* oraz technologii ODBC, pożądane jest, aby te obiekty mogły przechowywać niezbędne informacje dotyczące konkretnych baz danych zainstalowanych w systemach plików na pewnych serwerach sieciowych. Informacjami tymi są:

- **Alias** (nazwa) pewnej bazy danych, ustalany i konfigurowany przez program zarządzający *Borland Database Engine Configuration Utility*, za pośrednictwem którego realizowany jest dostęp do wskazanej bazy danych,
- **lista plików** bazodanowych do których zrealizowany ma być dostęp w celu wykonania prostych czynności zarządzających, tj. nawigacji po rekordach, edycji wyróżnionych pól informacyjnych, jak również dodawania nowych oraz kasowania wskazanych rekordów.

Ze względu na fakt, że konkretne pliki składające się na pewną bazę danych mogą rezydować w różnych podkatalogach, zdecydowano się, że wszystkie pliki będące na liście plików zawierać będą również pełną ścieżkę dostępu do nich. Wśród innych wymagań dotyczących tej nowej klasy obiektowej jest wymóg zapewnienia bezpieczeństwa w dostępie do informacji zawartych w plikach składowych baz danych. Innymi słowy, chodzi o to, aby wystąpienie tej klasy zawierało listę użytkowników, którzy mają mieć do nich dostęp. Biorąc pod uwagę powyższe rozważania, po analizie wszystkich klas obiektowych przedstawionych na rys. 12, które zdefiniowano w schemacie podstawowym NDS, najbardziej odpowiednią klasą, która mogłaby być podstawą do wyprowadzenia nowej - zdolnej sprostać narzuconym wymaganiom - jest klasa obiektowa *Server*. Na jej podstawie wyprowadzono nową klasę obiektową - *DBServer* - której kompletną definicję pokazano na rys. 13. Jak widać na tym rysunku, klasa *DBServer* dziedziczy wszystkie atrybuty opcjonalne z klasy *Server*. Ponadto dodano jeden

atrybut opcjonalny - *Database Files*. Reasumując, w celu realizacji narzuconych wymagań przyjęto następujące znaczenie niektórych atrybutów z projektowanej klasy obiektowej *DBServer*. I tak atrybut o nazwie:

- *Description* - przechowuje Alias do pewnej bazy danych,
- *Database Files* - przechowuje listę plików (tablic) wskazanej przez Alias bazy danych,
- *User* - przechowuje użytkowników mających dostęp do tej bazy danych.



Rys. 12. Hierarchia klas obiektowych zdefiniowanych w schemacie podstawowym
Fig. 12. Layout of the object classes definied in base schema

Klasa	Top	Server	DBServer
Nadklasa	*(brak)*	*Top*	*Server* *Top*
Zawiranie:	*(brak)*	*Organization* *Organizational Unit*	* *Organization* * *Organizational Unit*
Nazwany przez:	*(brak)*	*CN*	* *CN*
Atrybuty obligatoryjne	*Object Class*	* *Object Class* *CN*	* *Object Class* * *CN*
Atrybuty opcjonalne	ACL Back Link Bindery Property References Revision	* ACL * Back Link * Bindery Property * References * Revision Description Full Name Host Device Minimum Account Balance Network Address O OU Private Key Public Key Resource Security Equals Security Flags See Also Status User Version	* ACL * Back Link * Bindery Property * References * Revision * Description * Full Name * Host Device * Minimum Account Balance * Network Address * O * OU * Private Key * Public Key * Resource * Security Equals * Security Flags * See Also * Status * User * Version Database Files

Rys. 13. Definicja klasy *DBServer* - dziedziczenie własności z nadklas
Fig. 13. The *DBServer* class definition - property inheritance from superclasses

Praktyczną implementację tak zdefiniowanej nowej klasy obiektowej należy dokonać w podprogramie uruchamianym z poziomu rozszerzonego interfejsu użytkownika.

2.3. Wykonanie sparametryzowanej obiektowej aplikacji

Po naciśnięciu klawisza *Edytuj*, pokazanego na rys. 6, uruchamiana jest sparametryzowana obiektowa aplikacja windows'owa tworząca wystąpienie klasy *TDBWindow*, którą wyprowadzono z klasy *Twindow* (z biblioteki *Object Windows Library*).

```
class TDBWindow : public TWindow, public TVbxEventHandler
{
  public:
    TDBWindow(TWindow *A_Parent, char *A_Title, string A_DbName, string A_DbTable );
    virtual void SetupWindow();
    protected:
    string MyDatabaseName;
    string MyDatabaseTable;
    TDBDataSource *DBDataSource;   /* Klasy typu TVbxControl z biblioteki VDBT   */
    TDBTable      *DBTable;        /* zapewniające dostęp do danych.             */
    TDBGrid       *DBGrid;
    TDBNavigator  *DBNavigator;
    DECLARE_RESPONSE_TABLE( TDBWindow );
};
DEFINE_RESPONSE_TABLE2( TDBWindow , TWindow , TVbxEventHandler )
END_RESPONSE_TABLE;

TDBWindow::TDBWindow(TWindow *A_Parent, char *A_Title, string A_DbName, string A_DbTable) : TWindow( A_Parent , A_Title )
{
  SetBkgndColor( TColor(0, 0, 255) );
  Attr.X = 10; Attr.Y = 10; Attr.W = 520; Attr.H = 350;     /* ustalenie wymiarów okna */
  MyDatabaseName = A_DbName;
  MyDatabaseTable = A_DbTable;
  DBDataSource    = new TDBDataSource(this, 1100, "Test DBDataSource");
  DBTable         = new TDBTable(this, 1200,"Test TDBTable");
  DBGrid          = new TDBGrid(this, 1000, "Test TDBGrid", Attr.X + 10, Attr.Y + 10, Attr.W - 40, Attr.H - 100, 0);
  DBNavigator     = new TDBNavigator(this, 103,"", Attr.X + 10, Attr.H - 70, Attr.W - 40, 15);
}

void TDBWindow::SetupWindow()
{
  TWindow::SetupWindow();
  try {
       DBTable->TableName    = MyDatabaseTable;
       DBTable->DatabaseName = MyDatabaseName;
       DBDataSource->DataSet = DBTable;
       DBNavigator->DataSource = DBDataSource;
       DBGrid->DataSource    = DBDataSource;
       DBTable->Open();
       DBTable->TDataSet::Refresh();
     }
  catch (BDTException e)
    { e.Show("Blad"); }
}
```

Rys.14. Klasa obiektowa ***TDBWindow***, wyprowadzona z klasy *TWindow*
Fig. 14. The TDBWindow object class derived from TWindow class

Efekt działania tej klasy przedstawiono na rys. 7, natomiast jej wydruk przedstawiono na rys.14. Konstruktor tej klasy tworzy wystąpienia klas typu *TVbxControl* (z biblioteki VDBT):

- *TDBDataSource*

- *TDBTable*,
- *TDBGrid*,
- *DBNavigator*,

zadaniem których jest uzyskanie dostępu - wykorzystując zaimplementowane w nich mechanizmy *Borland Database Engine* oraz technologii ODBC - do wskazanego pliku bazy danych oraz poddanie go edycji.

3. Utworzenie oraz eksploatacja jednolitego środowiska sieciowego

3.1. Opis systemu informatycznego w ZTS „NITRON" S.A.

Obecnie eksploatowany system informatyczny z ZTS „Nitron" S.A. działający na osnowie sieci światłowodowej FDDI, pracuje pod kontrolą sieciowego systemu operacyjnego Novell NetWare 4.1, w skład którego wchodzą trzy serwery sieciowe, na których zainstalowano typowe systemy informacyjne („*Wyroby Gotowe*", „*Rozrachunki z klientami*", systemy finansowo-księgowe, system „*Kadrowo-Płacowy*" itd.) służące do zarządzania firmą. Część tych systemów działa na bazie plików typu *dBASEIII*, pozostałe zaś na bazie plików typu *Btrieve*. Zarządzanie aktualnymi zasobami sieciowymi dokonywane jest za pomocą programu *NetWare Administrator* uruchamianego na stacji roboczej w środowisku Windows 3.11 lub Windows 95.

3.2. Instalacja i konfiguracja utworzonych bibliotek .DLL typu Snapin

Utworzone według proponowanej metody biblioteki .DLL typu *snapin* zawierające niezbędny kod do tworzenia opisanej nowej klasy obiektowej (*dbclass.dll*) oraz kod do obsługi jej wystąpień (*dbserver.dll*) wywoływane są oraz inicjalizowane przez program *NetWare Administrator* z tzw. katalogu domowego (*home directory*) użytkownika, którego zadaniem jest dokonanie czynności naprawczych w bazach danych użytkowych systemów informacyjnych. Stanowi to jeszcze jeden, tj. czwarty element ochrony (ponad te które opisano w rozdziale 2) przed dostępem osób niepowołanych do informacji zawartych w plikach składowych baz danych. Użytkownicy, którzy nie mają dostępu do tych bibliotek, nie mają również dostępu do nowych klas obiektowych i ich wystąpień, w efekcie czego nie mają również dostępu do tych informacji. W przypadku użytkownika pełniącego rolę administratora sieci komputerowej, biblioteki te mogą rezydować w katalogu systemowym (SYS:SYSTEM), do którego nikt - poza nim (w przypadku prawidłowo nadanym użytkownikom praw do systemu plików [14]) - nie ma dostępu. Po skonfigurowaniu pozostałych elementów

środowiska sieciowego, tj. sterowników ODBC za pomocą programu zarządzającego *Microsoft ODBC Administrator* oraz motoru Borlanda (*Borland Database Engine - BDE*) za pomocą aplikacji *Borland Database Engine Configuration Utility*, środowisko dla administratora sieci lub uprawnionych użytkowników jest gotowe, aby wykorzystać możliwości oferowane przez te biblioteki (co przedstawiono na rysunkach 3-8), integrując w ten sposób różne zasoby sieciowe w jednolite i zarazem proste w zarządzaniu oraz wykorzystaniu środowisko sieciowe.

4. Podsumowanie

NDS będąca obiektowo zorientowaną implementacją usług katalogowych umożliwiającą hierarchiczne przedstawienie sieci komputerowej i jej zasobów, jako część systemu operacyjnego NetWare 4.x zyskała sobie od czasu swojej promocji w 1993 r. uznanie i dużą popularność. Jedną z najważniejszych cech NDS'u jest możliwość rozszerzenia schematu za pomocą łatwych w użyciu funkcji API (*Application Program Interface*), które umożliwiają dodanie do schematu nowych klas obiektowych. Dostępność w ramach SDK (*SoftWare Developer's Kit*) bogatego zestawu funkcji API oraz usług wspiera rozwój aplikacji i narzędzi wykorzystujących schematy rozszerzone o nowe klasy obiektowe. Jedną z takich usług jest tzw. *Snapin Services*, pozwalająca istniejącym narzędziom administracyjnym (*NetWare Administrator Utility*) wykorzystywać schematy rozszerzone. W niniejszym artykule zaproponowano metodę, która po wdrożeniu - oprócz rozwiązania omawianego problemu - daje następujące korzyści:

- umożliwia ujednolicenie problemu zarządzania dotychczasowymi zasobami sieciowymi oraz użytkowymi informacyjnymi bazami danych traktowanych jako zasoby sieciowe,
- wspólny interfejs dostępu do danych zawartych w plikach składowych baz danych ułatwia uprawnionym użytkownikom na dokonanie czynności naprawczych. Bogaty zestaw istniejących na rynku sterowników ODBC pozwala praktycznie na uzyskanie dostępu do każdej bazy zainstalowanej na serwerach sieciowych,
- wszelkie operacje na zasobach sieciowych dokonywane są z jednego miejsca za pomocą jednego wspólnego interfejsu użytkownika,
- ochrona informacji zawartych w plikach składowych baz danych.

LITERATURA

1. Herbon G.B.: An introduction to NetWare Directory Services. Application Notes, April 1993.
2. Understanding NetWare Directory Services. NetWare 4.1 Manual Set Chapter 2.
3. Trocino R.: Understanding Relational Theory. Applications Notes, July 1993.
4. Parker K.: Using NetWare SQL to Access Relational Information from Btrieve Data.Novell Applications Notes, September 1992.
5. Btrieve 386 Instalation and Operation. Novell Development Products Division, December 1989.
6. Bok Z.: Analiza modelu logicznego obiektowo zorientowanej bazy NDS w sieciowym systemie operacyjnym Novell NetWare 4.1. Software nr 3, marzec 1998.
7. Bunnel K.L.: Integrating with the NetWare Administrator Utility. Novell Developer Support, Revision 1.o, August, 1996.
8. Buckle J.: Incorporating NDS Schema Extensions into the NetWare Administrator Application. Novell Developer Support, November 1996.
9. Novell Technical Information Document: NWAdmin Snapin DLL Path Load Option. Document ID: TID100660 rev. A, 07/29/1996.
10. Fisher L. V.: Anatomia prostej aplikacji klient/server do IntranetWare (cz. II). Software, nr 12, grudzień 1997.
11. plik internetowy: Extending the NDS Schema Object Class and Attribute Definitions. http://developer.novell.com/schema.htm.
12. Managing NDS Entries. Novell Support Connection, version 1.3.JS, April 1997.
13. Borland C++ Programmer's Guide ver. 5.0. Borland International, Inc., 1996.
14. NetWare Administration Student Manual. Course 520, rev 1.02.
15. Moncur M., Chellis J.: IntranetWare. Warszawa, listopad 1997, ISBN 83-7158-070-3.
16. NetWare v.3.11 to 4.0 Update, Student Manual. Course 526, rev 1.0.
17. Introduction to NetWare Directory Services. Novell Support Connection, NetWare 4.1 Manual Set, Novell Inc, 1996.
18. NDS Technical Overview. Novell Support Connection, version 1.3.JS, April 1997.
19. Coad P., Yourdon E.: Object-Oriented Design. Prentice Hall Inc., 1991, ISBN 83-85769-18-8.
20. Mak M.: Using NDS To Make a Service Globally Accessible. Novell Developer Notes, April 1996.

21. Subieta K.: Słownik często spotykanych terminów dotyczących obiektowości. Instytut Podstaw Informatyki PAN, Warszawa, wrzesień 1997, http://www.ipipan.waw.pl./~subieta.
22. Atkinson M., Bancilhon F., DeWitt D. Dittrich K., Maier D., Zdonik S.: The Object-Oriented Database System Manifesto. In Proceedings of the First International Conference on Deductive and Object-Oriented Databases. Kyoto, Japan, December 1989, http://www.cs.cmu.edu/People/clamen/OODBMS/Manifesto/htManifesto/Manifesto.html.
23. Barry D.K.: The Object Database Handbook. John Wilwy & Sons, Inc., USA, 1996, ISBN 0-471-14718-4.
24. Cattel R.: The Object Database Standard: ODMG-93. Release 1.1, San Mateo, CA: Morgan Kaufmann, 1994.
25. Loomis Mary E.S.: Object Databases. The Essentials. Addison-Wesley Publishung Company, 1995, ISBN 0-201-56341-X.

Recenzent: Dr hab. inż. Henryk Rybiński

Wpłynęło do Redakcji 30 czerwca 1998 r.

Abstract

In this article - on example industrial information system working in ZTS 'Nitron' S.A. factory under network Novell NetWare 4.x operating system control, an access method to information databases based on object-oriented database NDS and ODBC technology was presented. Based on this method a resource management process such as users, file servers, printers etc. together with industrial information management systems that controls *dBaseIII* and *Btrieve* type databases installed in file systems on network file servers has been integrated into uniform under management network environment.

This method on the following assumption is based, that if every network resource in NetWare 4.x network is represented by NDS object which was created and managed by user's interface called NetWare Administrator, then an other objects represented different network resources (as presented on fig. 4, 5) can be created in NDS too. These objects with new properties can be linked with certain databases installed on certain network file servers that allows a direct access (as presented on fig. 7) to them by using ODBC technology.