STUDIA INFORMATICA 2002
Volume 23 Number 3 (50)

Marek MITTMANN
Politechnika Śląska, Instytut Informatyki

WIELOWARSTWOWA ARCHITEKTURA DOSTĘPU DO DANYCH Z PROGRAMOWALNYMI WIDOKAMI

Streszczenie. Przedstawiona w artykule biblioteka i współpracujące z nią programy narzędziowe powstały w ramach prac mających na celu stworzenie uniwersalnych rozwiązań ułatwiających budowanie aplikacji wielostanowiskowych korzystających z bazy danych. Biblioteka zawiera mechanizmy realizujące: komunikację pomiędzy warstwami, sterowanie przepływem danych, obsługę wielu użytkowników, ochronę danych, zarządzanie uprawnieniami i interpretację skryptów konfiguracyjnych. Jednak najistotniejsze jest wprowadzenie warstwy obsługującej wirtualne, programowalne widoki danych, pozwalające na uniezależnienie logiki aplikacji od fizycznej struktury bazy danych.

MULTI-TIERED DATA ACCESS ARCHITECTURE WITH PROGRAMMABLE VIEWS

Summary. A library and tools presented in the paper has been created as a part of the research on the universal solutions for creating multi-tiered database applications. The library provides: communication between tiers, data flow control, multi-user support, data protection, managing of the access rights and scripts interpretation. However, the most important is support for the tier that provides virtual programmable data views which make application logic independent of a physical database structure.

1. Wstęp

Potrzeba ciągłego zwiększania zakresu funkcjonalności oprogramowania użytkowego jest bezdyskusyjna. Często wykorzystuje się w tym celu zaawansowane technologie i gotowe biblioteki, które zwiększając możliwości tworzonego oprogramowania jednocześnie skracają

czas ich projektowania. Wiąże się z tym również dobór odpowiedniej architektury. W przypadku aplikacji wielostanowiskowych rozdziela się warstwę danych i przetwarzania, uzyskując rozwiązania typu klient-serwer lub architekturę wielowarstwową. Rozdzielenie to nie jest jednak pełne: architektura aplikacji jest ściśle uzależniona od logicznej struktury użytkowanej bazy danych. Stanowi to istotne ograniczenie rozwoju aplikacji, szczególnie w razie konieczności wprowadzenia istotnych zmian bądź to po stronie aplikacji, bądź też bazy danych.

Przedstawione w artykule rozwiązanie stanowi propozycję architektury aplikacji wielostanowiskowych, w których logika aplikacji może zostać całkowicie uniezależniona od faktycznej struktury bazy danych. Uzyskuje się to poprzez wprowadzenie dodatkowej warstwy wirtualnego, programowalnego widoku danych, która stanowi moduł pośredniczący pomiędzy aplikacją i jej bazą danych.

Bezpośrednim powodem powstania prezentowanego rozwiązania był jeszcze jeden, wcale nie tak rzadko spotykany problem: należało zaimplementować rozbudowany moduł aplikacyjny, przy czym u różnych odbiorców miał on pracować z bazami danych o różnej strukturze. Wprowadzenie zróżnicowanej, ale prostej warstwy pośredniej pozwoliło zamodelować identyczne warunki pracy modułu bez względu na różnice w faktycznej organizacji danych.

Proponowane rozwiązanie obejmuje bazującą na technologii COM [1 – 4] bibliotekę wraz z oprogramowaniem narzędziowym (w tym środowiskiem projektowania dla specjalnego języka skryptowego, wykorzystywanego do definiowania widoków danych).*

2. Realizacja

Opisywane mechanizmy są niczym innym jak jednym ze sposobów na zorganizowanie aplikacji opartej na architekturze trójwarstwowej [5 - 7]. Ich struktura jest przedstawiona na rysunku 1. Większość zaprojektowanych elementów jest umieszczona w serwerze aplikacji. Warstwa prezentacji, zwana również klientem, zawiera składniki wspomagające tworzenie interfejsu użytkownika. Za bazę danych może służyć dowolny serwer danych wspierający język *SQL*, obsługiwany przez mechanizmy *BDE* lub *ODBC* [8, 9].

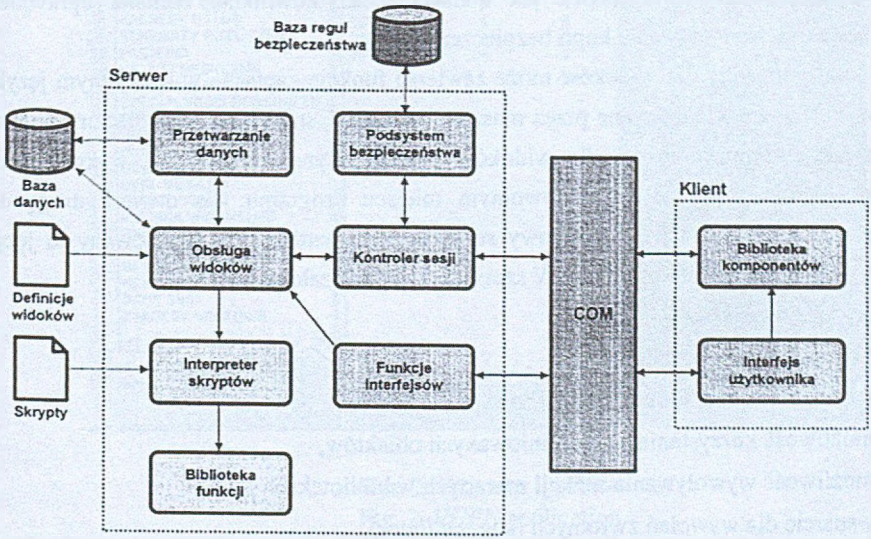
Komunikacja serwera z klientem została oparta na technologii *COM* [1 – 4]. Dzięki niej, stosunkowo niewielkim kosztem, zrealizowana została zarówno łączność lokalna, jak i komunikacja poprzez sieć komputerową. Funkcjonalność *COM* okazała się też niezwykle pomocna przy tworzeniu podsystemów bezpieczeństwa i zarządzania sesjami.

* Praca powstała dzięki dofinansowaniu w ramach Badań Własnych za rok 2002

Dane udostępniane przez serwer aplikacji zorganizowane zostały w logiczną strukturę, niezależną od faktycznego sposobu ich przechowywania i rozmieszczenia w bazie. Podstawą tej organizacji są tak zwane widoki. Dla użytkownika widoki są ciągiem rekordów, które z kolei złożone są z pól różnego typu, wypełnionych wartościami. Jednak w praktyce nie przechowują one żadnych danych, a jedynie określają sposób ich prezentacji i edycji. Na widokach można wykonywać następujące podstawowe operacje:

- zmiana aktywnego rekordu,
- pobranie danych z aktywnego rekordu,
- dodanie rekordu,
- usunięcie rekordu,
- edycja.

Sposób realizacji tych operacji definiuje się indywidualnie dla każdego widoku w postaci szeregu zapytań języka *SQL*. Stworzono też możliwość tworzenia procedur w specjalnym języku skryptowym *Pascal Script* [10], za pomocą których można przeprowadzać dodatkowe operacje na danych. Procedury te mogą być wykorzystane między innymi do kontroli poprawności danych w trakcie edycji. Opis struktury widoków oraz definicje wykonywanych na nich operacji są przechowywane w pliku konfiguracyjnym serwera aplikacji.



Rys. 1. Schemat architektury mechanizmów dostępu do danych
Fig. 1. Architecture of the data access mechanisms

Jednym z najistotniejszych elementów serwera jest kontroler sesji. Część jego metod jest dostępna na zewnątrz za pośrednictwem interfejsu *IKMConnect*. Dla każdego połączenia klienta z serwerem tworzony jest jeden kontroler sesji. Jego rola polega na inicjowaniu połączenia z użytkownikiem i udostępnianiu pozostałych interfejsów. Jest to jedyny obiekt dostępny dla klienta przed rejestracją. Rejestracja polega na przesłaniu identyfikatora użytkownika i prawidłowego hasła za pomocą odpowiedniej funkcji *IKMConnect*. Informacje te są przekazywane do podsystemu bezpieczeństwa, który sprawdza, czy dany użytkownik znajduje się w bazie reguł bezpieczeństwa. Jeżeli wynik jest pozytywny, sesja przechodzi w stan pełnej aktywności i odtąd klient może kierować żądania dostępu do pozostałych interfejsów. Oczywiście, każde takie żądanie poddawane jest ocenie podsystemu bezpieczeństwa, której rezultat wynika z uprawnień użytkownika.

W bloku opisanym jako obsługa widoków znajdują się obiekty sterujące pobieraniem danych z bazy, ich przetwarzaniem i formowaniem do postaci dostępnej na zewnątrz. Tu interpretuje się definicje odczytane z pliku konfiguracyjnego oraz wykonuje wszystkie podstawowe operacje związane z widokami. Dostęp do widoków z zewnątrz jest realizowany za pomocą interfejsu *IKMDataSet*. Zawiera on zestaw funkcji pozwalających na przemieszczanie pomiędzy rekordami, pobieranie i edycję danych oraz pobieranie informacji o strukturze i stanie widoku.

Kolejnym z dostępnych interfejsów jest *IKMAdmin*. Jego funkcje służą do wykonywania czynności administracyjnych, takich jak dodawanie użytkowników, zmiana uprawnień, zmiana hasła oraz wykonywanie kopii bezpieczeństwa.

Plik konfiguracyjny dla widoków może zawierać funkcje zapisane w specjalnym języku skryptowym. Są one wykonywane przez maszynę wirtualną, stanowiącą nieodłączny element opisywanego systemu. Funkcje dla widoków nie są jedyną możliwością wykorzystania skryptów. Można stosować je w dowolnym miejscu programu. To otwiera drogę dla zaawansowanej konfiguracji i rozbudowy serwera. Tym bardziej że zastosowany tu język skryptowy ma spore możliwości [10]. W szczególności należałoby wymienić:

- obsługę wyjątków,
- różnorodność typów danych,
- bogatą bibliotekę funkcji standardowych,
- możliwość korzystania z predefiniowanych obiektów,
- możliwość wywoływania funkcji zawartych w bibliotekach *DLL*,
- wsparcie dla wywołań zwrotnych (ang. *callback*),
- podobieństwo do języka Pascal.

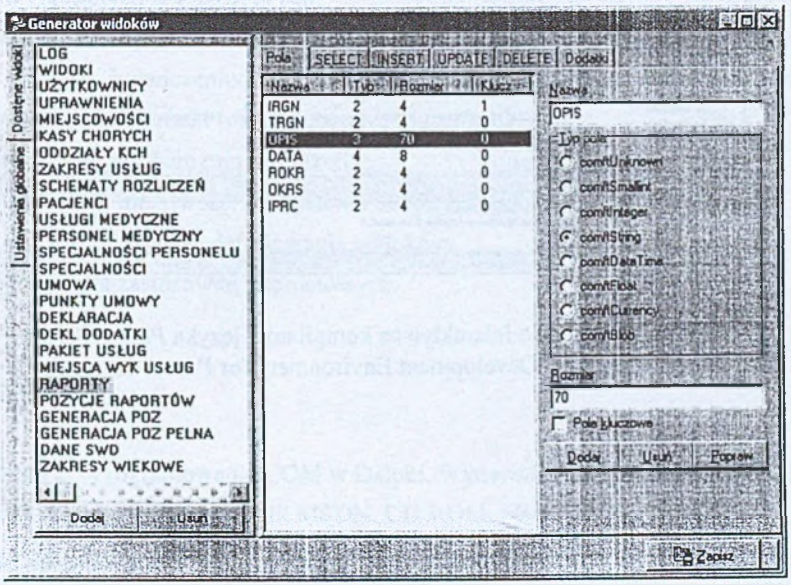
Warstwa prezentacji może komunikować się z serwerem za pośrednictwem samych tylko interfejsów *COM*. Jednak w przypadku, gdy klient jest tworzony za pomocą kompilatorów

wyposażonych w bibliotekę *VCL* [11], można wykorzystać zestaw komponentów, które pozwalają na integrację tej biblioteki z prezentowanymi mechanizmami. Dzięki temu zabiegowi przy tworzeniu interfejsu można w pełni wykorzystywać wszystkie udogodnienia wnoszone przez kompilatory korzystające z *VCL*.

3. Programy narzędziowe

Jednym z mankamentów prezentowanego rozwiązania jest konieczność definiowania struktury danych udostępnianych przez serwer. Ręczna edycja pliku konfiguracyjnego jest bardzo niewygodna i pracochłonna. Dlatego stworzony został pakiet interaktywnych narzędzi wspomagających przeprowadzanie tej operacji.

Podstawowym narzędziem używanym do konfiguracji jest *VGen* (rys. 2). Program ten pozwala na wygodne definiowanie, przeglądanie i modyfikowanie widoków.

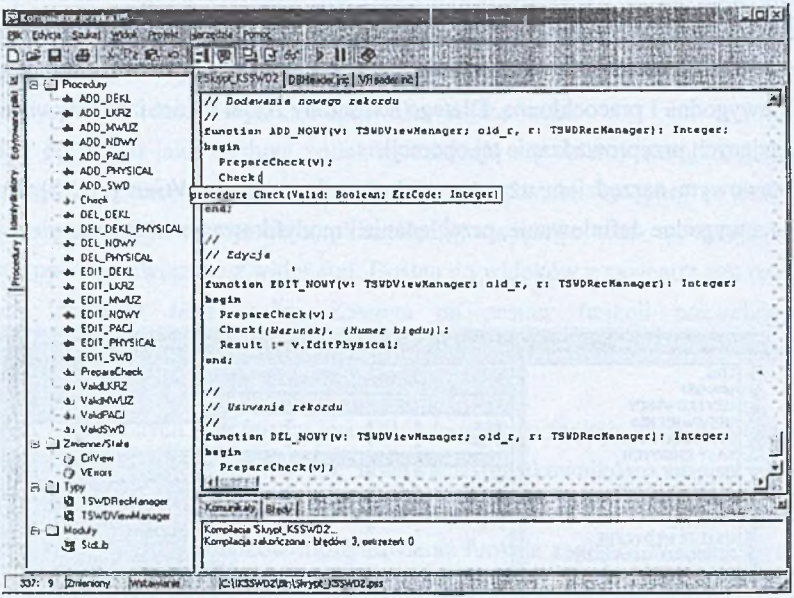


Rys. 2. Program *VGEN*
Fig. 2. *VGEN* application

Najciekawszym składnikiem pakietu jest kompilator języka *Pascal Script*, wyposażony w interaktywne środowisko z nowoczesnym edytorem i narzędziami wspomagającymi tworzenie skryptów (rys. 3). Za jego pomocą można przygotowywać i testować skrypty dla

serwera aplikacji. Dostępne są takie udogodnienia jak: kolorowanie składni, wyświetlanie podpowiedzi, przemieszczanie do wybranych fragmentów kodu, wskazywanie błędów składniowych oraz przeróżne kreatory.

Kompilator oraz maszyna wirtualna języka *Pascal Script* mogą być też używane niezależnie od mechanizmów dostępu do danych. Pozwalają na kompilowanie i uruchamianie programów w sposób analogiczny do zastosowanego dla *Javy*.



Rys. 3. Środowisko interaktywne kompilatora języka *Pascal Script*
Fig. 3. Integrated Development Environment for *Pascal Script*

4. Podsumowanie

Wynikiem prac nad realizacją opisywanych mechanizmów jest **biblioteka** stanowiąca szkielet **wielostanowiskowej aplikacji** oraz pakiet programów narzędziowych. Rozwiązanie to oparte jest na **technologii COM**, dzięki czemu może być wykorzystywane w większości programów pracujących pod kontrolą systemu operacyjnego *Windows*. Użytkownicy kompilatorów *Delphi* i *C++ Builder* mogą korzystać z dodatkowego wsparcia w postaci **komponentów** ułatwiających implementowanie warstwy prezentacji.

Największą korzyścią wnoszoną przez prezentowaną bibliotekę jest możliwość jednoczesnego korzystania i przetwarzania wspólnych danych przez **wielu użytkowników**,

pracujących na komputerach rozlokowanych w dowolnym miejscu lokalnej sieci. Dużo pracy włożono też w wykonanie elementów odpowiedzialnych za **bezpieczeństwo danych**. Zastosowano uznaniową kontrolę dostępu – każdemu użytkownikowi można nadawać uprawnienia pozwalające na dostęp do poszczególnych informacji. Poziom bezpieczeństwa odpowiada klasie C2.

Jednym z najistotniejszych elementów są mechanizmy udostępniania i prezentacji danych w sposób niezależny od ich faktycznej struktury w bazie, które można by określić mianem „czwartej warstwy", rozwijającej model trójwarstwowy. Podstawowe składniki struktury danych dostarczanych przez serwer aplikacji noszą nazwę **widoków**. Definicje widoków mogą być uzupełnione o procedury weryfikujące i przetwarzające dane. Do zapisu tych procedur stworzono język *Pascal Script*, którego można również używać do tworzenia skryptów konfiguracyjnych i makrodefinicji.

Dołączone do biblioteki programy wspomagają definiowanie widoków. Na uwagę zasługuje *kompilator języka Pascal Script*, który wraz z *maszyną wirtualną Pascal Script* tworzy niezależny pakiet, pozwalający na tworzenie i uruchamianie w systemie *Windows* programów zapisanych w *Pascal Script*.

Opisywane mechanizmy zostały z sukcesem zaimplementowane w jednej z wersji systemu obsługi świadczeniodawców Kas Chorych KS-SWD firmy Kamsoft [10]. Potencjalne przyszłe kierunki rozwoju tego rozwiązania obejmują:

- rozwinięcie mechanizmu transakcji,
- rozszerzenie możliwości podsystemu bezpieczeństwa,
- uproszczenie procesu definiowania widoków,
- wsparcie dla zastosowań internetowych.

LITERATURA

1. Harmon E.: Programowanie COM w Delphi. Wydawnictwo Translator, 2000.
2. Microsoft Developers Network MSDN. CD-ROM, Microsoft Corp. 2000.
3. Rogerson D.: Inside COM. Microsoft Press, Redmond 1997.
4. Eddon G., Eddon H.: Inside Distributed COM. Microsoft Press, Redmond 1998.
5. N-tier Computing Architecture. Artykuł zamieszczony w serwisie internetowym Delaware.gov, http://www.state.de.us/ois/arch/n-tier.html.
6. Parkes C. H.: In Search of Great Architecture. Artykuł zamieszczony w serwisie DBMS Online, luty 1998, http://www.dbmsmag.com/9802d19.html.

7. Inprise Application Server - Zintegrowane rozwiązanie do tworzenia, wdrażania i zarządzania wielowarstwowymi rozproszonymi aplikacjami. Artykuł zamieszczony w serwisie Borland Polska, http://www.bsc.com.pl/tech/.
8. Borland Database Engine Reference. Inprise Corporation, 1999.
9. Zarzycki Z.: Porównanie różnych metod dostępu do baz danych. Artykuł zamieszczony w serwisie Borland Polska, http://www.bsc.com.pl/tech/.
10. Mittmann M.: Projekt i realizacja mechanizmu kontroli dostępu do danych w programie KS-SWD. Praca dyplomowa magisterska, Instytut Informatyki, Politechnika Śląska 2001.
11. Visual Component Library Reference. Inprise Corporation, 1999.

Recenzent: Dr inż. Arkadiusz Sochan

Wpłynęło do Redakcji 8 kwietnia 2002 r.

Abstract

A library presented in the paper provides support for creating multi-tiered database applications. This is based on the COM technology. Figure 1 shows an internal structure of the library. The Application Server contains three COM interfaces: *IKMConnect*, *IKMDataSet* and *IKMAdmin*, which can be used to access data. *Delphi* and *C++ Builder* users can use the extra library with components that support creating the user interface. The Server needs definition of the logical data structure. This can be made using *VGEN* (fig. 2). The logical data structure makes an application logic independent of a physical database structure.

The *Pascal Script Compiler* (fig. 3) was created for writing and testing scripts which can be run on the application server.