ISSN 1507-3858
e-ISSN 2450-0003

Sebastian Łacheciński
Uniwersytet Łódzki
e-mail: slachecinski@uni.lodz.pl

INTERFEJSY DOSTĘPU DO BAZ DANYCH – PRZEGLĄD TECHNOLOGII MICROSOFT

INTERFACES OF ACCESS TO DATABASES – MICROSOFT TECHNOLOGY REVIEW

DOI: 10.15611/ie.2015.3.05
JEL Classification: C88, L63, L86

Streszczenie: Artykuł poświęcono niezwykle istotnemu zagadnieniu, jakim jest realizacja dostępu aplikacji do różnych źródeł danych. Przedstawia on rozwój, który nastąpił w ostatnim ćwierćwieczu w odniesieniu do najbardziej popularnych interfejsów dostępu do danych. Dzięki nim realizowany jest dostęp aplikacji do danych przechowywanych i zarządzanych przez różne serwery SQL. Zaprezentowano w nim architekturę wybranych interfejsów opracowanych przez Microsoft. Celem artykułu jest wyznaczenie użyteczności analizowanych interfejsów z jednoczesnym wskazaniem wad i zalet poszczególnych technologii. Artykuł ten jest swoistą analizą porównawczą prezentowanych w nim interfejsów.

Słowa kluczowe: interfejs dostępu do danych, ODBC, OLE DB, DAO, RDO, RDS, ADO, ADO.NET.

Summary: This article is devoted to an extremely important issue, which is the implementation of applications access to various data sources. It shows the development that took place during the last quarter of a century for the most popular interfaces of access to data. Interfaces implement the access of applications to data stored and managed by different SQL servers. The paper presents the architecture of selected interfaces developed by Microsoft. The aim of the article is to determine the usefulness of the analyzed interfaces while indicating the advantages and disadvantages of each technology. This article is a specific comparative analysis of presented interfaces.

Keywords: interface of access to databases, ODBC, DAO, RDO, OLE DB, ADO, RDS, ADO.NET.

1. Wstęp

Praktycznie każdy współczesny system informatyczny korzysta bezpośrednio czy też pośrednio współpracuje z różnymi formami źródeł danych, poczynając od szerokiej gamy systemów bankowych, przez systemy zarządzania relacjami z klienta-

mi CRM, systemy zarządzania przedsiębiorstwem ERP, systemy wykorzystywane w logistyce do zarządzania łańcuchami dostaw SCM oraz systemy zarządzania procesami magazynowymi WMS, systemy planowania potrzeb materiałowych MRP, systemy planowania zasobów wytwórczych MRP II, systemy informacji geograficznej GIS, na zastosowaniach *stricte* inżynierskich czy też typowo naukowych bazach danych kończąc. Jak widać, spektrum zastosowania baz danych jest praktycznie nieskończone. Niezależnie od rodzaju użytego źródła danych − czy to w postaci plików tekstowych, xml, czy też baz danych opartych o różne modele danych, aby aplikacja mogła z nich korzystać i je przetwarzać − przede wszystkim należy zapewnić aplikacji dostęp do danych składowanych w konkretnym źródle. Zatem kluczową rolę odgrywa zapewnienie dostępu aplikacji klienta do źródła danych. Jednym z elementów składających się na architekturę aplikacji opierającej się na bazie danych, pełniącym rolę łącznika między bazą danych a samą aplikacją, jest odpowiednio zaimplementowany interfejs (bazujący na odpowiedniej metodzie dostępu do baz danych). W ostatnim ćwierćwieczu powstało wiele różnych metod dostępu do baz danych, które zapewniły aplikacjom dostęp do danych składowanych w różnych źródłach danych, m.in. również na serwerach SQL.

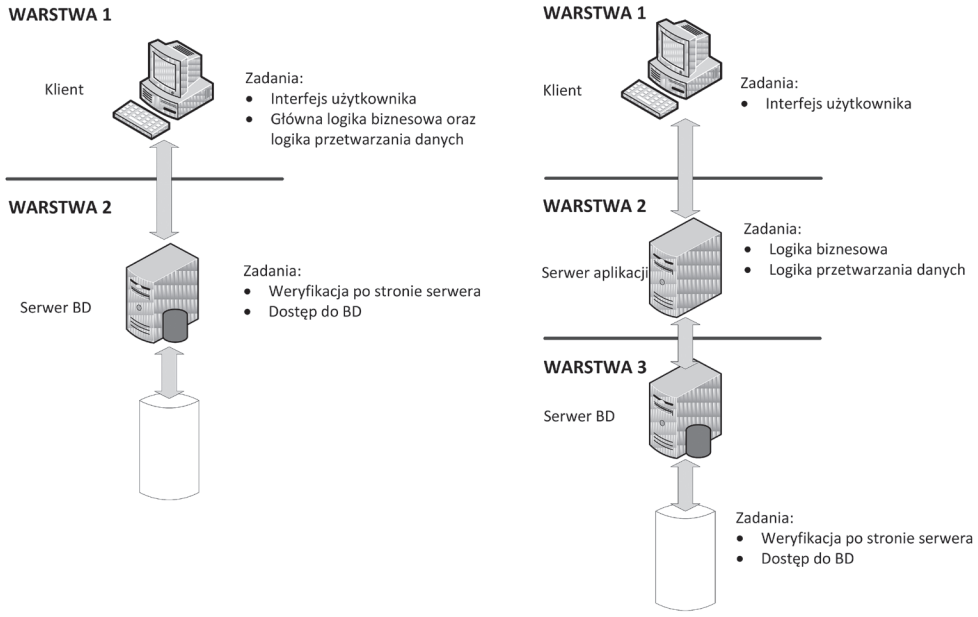
Interfejsy dostępu do baz danych stanowią zbiór funkcji umożliwiających łączenie i komunikowanie się aplikacji klienckiej z wybranymi SZBD. Zaimplementowane w postaci programu funkcje pozwalają na komunikację z bazą danych, wymianę danych, przekazywanie poleceń, konfigurację dostępu do danych itd., a co najważniejsze − wszystko to możliwe jest najczęściej bez konieczności gruntownej znajomości wiedzy na temat implementacji klienta i serwera. Stanowią one pomost, swojego rodzaju łącze programowe między aplikacją a bazą danych.

Rys. 1. Przykładowa wielowarstwowa architektura aplikacji bazodanowej

Źródło: opracowanie własne.

Opracowanie rozwiązań w postaci interfejsów dostępu do danych stanowiło przełom w projektowaniu aplikacji opierających się na bazach danych. Dzięki zastosowaniu

takiego podejścia jedna aplikacja za pomocą tych samych instrukcji uzyskała dostęp do wielu różnych baz danych. Podejście takie znacznie zwiększyło możliwości projektowania i ułatwiło budowanie aplikacji w architekturze klient-serwer o wszechstronnym zastosowaniu, niezwiązanych z konkretnym serwerem SQL. Na rysunku 1 zilustrowano przykładową wielowarstwową architekturę aplikacji bazodanowej.



Rys. 2. Podział składowych aplikacji bazodanowej między poszczególne warstwy w architekturze dwu- i trójwarstwowej

Źródło: opracowanie własne na podstawie [Connolly, Begg 2004].

Aplikacja korzystająca z zasobów bazodanowych składa się z kilku mechanizmów zapewniających jej prawidłowe funkcjonowanie. Wśród nich można wyróżnić [Darakhvelidze, Markov 2005]:

- Odbieranie i wysyłanie danych − jego rolą jest realizacja połączenia z bazą danych. Przy tym wymagana jest znajomość źródła danych, jak również protokołu transmisji w celu obsługi dwukierunkowego strumienia danych.
- Wewnętrzną reprezentację danych − przechowuje odebrane dane i rozsyła je do innych części aplikacji. Tym samym jest rdzeniem aplikacji bazodanowej.
- Logikę biznesową do przetwarzania danych − ma ona postać zbioru algorytmów przetwarzania danych przez aplikację. W zależności od liczby użytkowników, wielkości bazy danych, jak również technik użytych w celu nawiązania połączenia z bazą danych może być realizowany na kilka różnych sposobów (środowisko aplikacyjne, bez którego aplikacja nie działa, zbiór sterowników i bibliotek

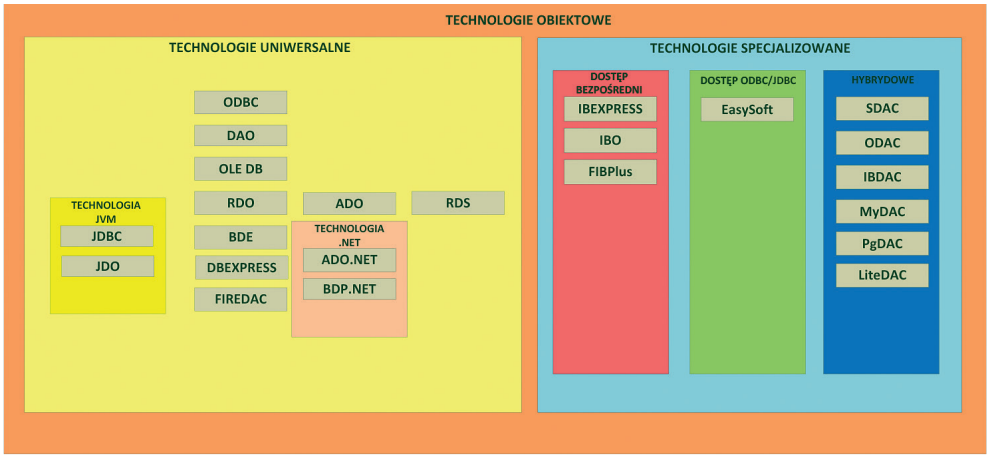
dynamicznych, do których odwołuje się aplikacja, zdalny serwer obsługujący dziesiątki tysięcy użytkowników).

- Interfejs użytkownika – pozwalający zarządzać aplikacją, przeglądać i edytować dane.

W zależności od zastosowanej architektury aplikacji poszczególne elementy aplikacji składowane są w różnych warstwach. Na rysunku 2 zaprezentowano przykładową dwu- i trójwarstwową architekturę aplikacji opartej na bazie danych [Connolly, Begg 2004].

2. Wybrane interfejsy

W ostatnich trzech dekadach powstało wiele różnych metod dostępu do danych i baz danych. Nad opracowaniem jednych, jak choćby standardu ODBC czy BDE, pracowało kilku gigantów z branży IT. W przypadku innych interfejsów były one opracowane przez pojedyncze firmy, m.in. dbExpress. Wybrane interfejsy dostępu do baz danych cechuje uniwersalność (ODBC, JDBC, FireDAC), podczas gdy drugie − specjalizowany dostęp do dedykowanego serwera baz danych, czego przykładem jest m.in. IBX, ODAC. Jedne z nich są wieloplatformowe (JDBC, FireDAC), zaś inne ukierunkowane są tylko na konkretny system operacyjny, w obrębie którego są w stanie działać (DAO, RDO, ADO). Nie bez znaczenia jest także sama technologia użyta do stworzenia aplikacji opierającej się na bazie danych, gdyż wraz z użytą technologią zmieniać się może metoda dostępu do danych [Barczak, Zacharczuk; Mościcki 2006]. Omówiony podział został przedstawiony na rys. 3.



Rys. 3. Klasyfikacja wybranych interfejsów dostępu do danych

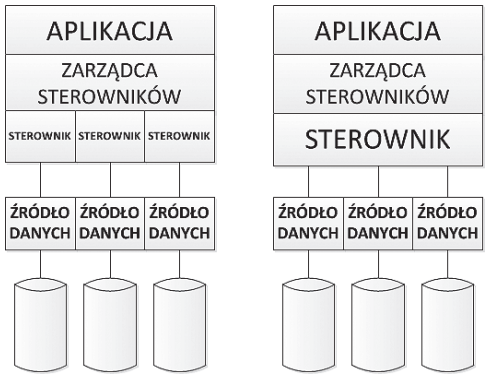
Źródło: opracowanie własne.

2.1. ODBC − *Open Database Connectivity*

Standard ten został opracowany we wrześniu 1992 r. przez SQL Access Group. [ADO ODBC OLE DB]. Składa się na niego zbiór funkcji API, dzięki którym możliwy stał się dostęp do relacyjnych baz danych. Dzięki niemu tworzone aplikacje mogły uzyskać dostęp do danych składowanych w plikach jak również w serwerach SQL [Roman 2001; Wybrańczyk 2004].

Niezwykle ważną kwestią, jak na tamte czasy, było to, iż standard ODBC był niezależny zarówno od języka programowania, systemu operacyjnego, jak i od samej bazy danych. Od tej pory jedna aplikacja uzyskała możliwość odwoływania się do wielu różnych baz danych za pomocą tych samych instrukcji. Tym samym aplikacja przestała być zależna od rzeczywistych protokołów dostępu do danych [Freeze 2001].

Standard ODBC zakłada, że procedury API, wywoływane przez aplikację, tłumaczone są na niższe wywołania przekazywane do sterownika bazy danych. Przy tym to właśnie sterownik jest odpowiedzialny za komunikację z serwerem baz danych. Dzięki takiemu rozwiązaniu producenci baz danych od tej pory dostarczali tylko sterownik kompatybilny z ODBC dla każdego systemu, który był klientem serwera baz danych (zamiast dla każdego kompilatora, w każdym systemie operacyjnym). Zadaniem producentów baz danych było dostarczenie interfejsu ODBC, który miał być następnie współużytkowany przez klientów ODBC [Connolly, Begg 2004].



Rys. 4. Architektura ODBC

Źródło: opracowanie własne na podstawie [Connolly, Begg 2004].

Architektura ODBC została zaprezentowana na rys. 4. Wyszczególnić można w niej cztery składowe:

1. Aplikację – której zadaniem jest przetwarzanie danych, jak również wywołanie funkcji ODBC, które przekażą polecenia SQL do konkretnego SZBD, a następnie zwrócą wyniki otrzymane od SZBD.

2. Zarządcę sterowników − zadaniem którego jest podłączanie i odłączanie sterowników dla aplikacji. Ponadto przetwarza on wywołania funkcji ODBC bądź też przekazuje je do sterownika.

3. Agenta sterowników i baz danych – zadaniem którego jest przetwarzanie wywołań funkcji ODBC, przekazywanie poleceń SQL do konkretnych źródeł danych i zwracanie wyników do aplikacji. Przy tym sterowniki realizują funkcje SZBD, z którymi są powiązane.

4. Źródła danych – które zawiera dane użytkownika, SZBD, w którym są one przechowywane, system operacyjny, jak również platformy sieciowej.

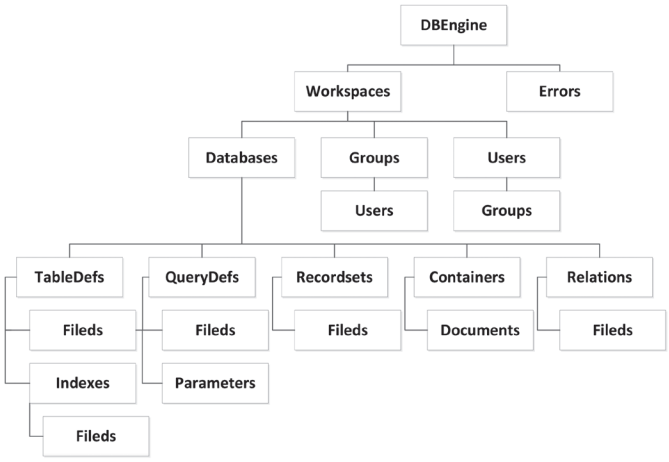
Do zalet ODBC zaliczyć można obecność we wszystkich wersjach systemu Windows, prostotę instalacji i łatwość konfiguracji, dostęp do wielu popularnych baz danych, jak również duża mobilność aplikacji. Wśród wad ODBC wymienić można: stosunkowo wolne połączenie z bazami danych (przez co nie nadaje się do zastosowań profesjonalnych), konieczność oddzielnych instalacji i konfiguracji komputerów, z których uzyskiwany będzie dostęp do baz danych, optymalizowany przez Microsoft dla baz danych typu Jet i MS SQL Server (w przypadku innych serwerów SQL warto rozważyć rozwiązania innych producentów oferujących większą wydajność jak i funkcjonalność).

2.2. DAO – *Data Access Objects*

Jest to programistyczny interfejs zorientowany obiektowo. Stworzony został głównie z myślą o dostępie do baz danych typu Jet z poziomu języka Visual Basic. Pierwsza wersja DAO 1.0 pojawiła się w listopadzie 1992 r. wraz z pojawieniem się pierwszej wersji Accessa. Stanowi programową nakładkę nałożoną na niskopoziomowy mechanizm wymiany danych ODBC, za pomocą której możliwa jest implementacja komunikacji między źródłem danych a aplikacją. Interfejs DAO oddziela logikę biznesową od dostępu do danych, dzięki czemu w łatwy sposób można zmienić źródło danych bez konieczności zmiany kodu aplikacji odpowiedzialnego za logikę biznesową [DAO; Roman 2001].

Model DAO ma strukturę hierarchiczną, co zilustrowano na rys. 5.

Podstawę hierarchii modelu danych stanowi silnik bazy danych. Przestrzenie robocze stanowią otoczenie dla otwieranych baz danych w ich obrębie (każda sesja reprezentowana jest przez jeden obiekt **Workspace**). Poziom bazy danych natomiast oferuje dostęp do definicji tabel, zapytań oraz zestawów rekordów. Dzięki temu możliwy jest dostęp do danych składowanych w tabelach. Obiekt **Relation** reprezentuje związki między tabelami i kwerendami. Obiekt **Users** reprezentuje użytkowników bazy danych, a obiekt **Group** reprezentuje zbiór obiektów **Users**, które mają wspólny zbiór uprawnień dostępu. Z kolei kolekcja obiektów **Errors** jest czyszczona i wypełniana obiektami **Error** dla każdego wystąpienia błędu. W obiektach **Container** aplikacja macierzysta przechowuje swoje obiekty. Motor Jet bazy danych tworzy kontenery dla obiektów: bazy danych, tabel, kwerend, związków, formularzy, raportów, makr i modułów [Pelikant 2009].



Rys. 5. Model DAO

Źródło: opracowanie własne na podstawie [Roman 2001].

Wśród zalet DAO wyróżnić można: niezależność implementacji aplikacji od sposobu składowania danych oraz większą niezależność przy łączeniu ze źródłem danych realizowaną przez zawarcie dostępu do danych w interfejsie. Wśród wad DAO wymienić zaś można głównie złożoność samego rozwiązania wynikającą ze zwiększenia liczby obiektów w systemie, która jest konsekwencją definiowania dostępu do obiektów DAO, fakt optymalizowania raczej na bazy plikowe oparte o motor Jet, tj. MS Access, Paradox, dBase.

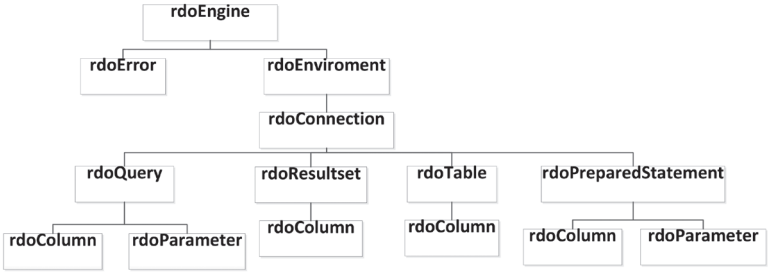
2.3. RDO – *Remote Data Objects*

Stanowi uporządkowany zbiór obiektów mających za zadanie ułatwić korzystanie z ODBC. Opracowany został w celu pozbycia się problemów, jakie pojawiały się podczas uzyskiwania dostępu do bazy MS SQL Server czy Visual Fox Pro. W przeciwieństwie do modelu DAO nie oferuje możliwości tworzenia bazy danych przez zdefiniowanie obiektów. RDO dołączane było do wersji 4, 5 i 6 Visual Basic-a. Finalną wersją była wersja 2.0. Ostatecznie RDO ewoluowało w ADO [RDO].

Na rysunku 6 przedstawiono hierarchiczny model RDO.

Interfejs RDO zaprojektowany został dla aplikacji pracujących pod kontrolą 32-bitowych systemów operacyjnych ze ściśle relacyjnymi bazami danych. W przeciwieństwie do DAO nie obsługuje wszelkiego rodzaju więzów integralności czy też mechanizmów bezpieczeństwa na rzecz narzędzi serwerowych, które to odpowiedzialne są za ich obsługę. Możliwe jest także wykonywanie procedur przechowywanych parametryzowanych bądź nieparametryzowanych, które mogą również zwracać wartość. Poszczególne obiekty prezentowane na modelu RDO odpowiadają określonym obiektom DAO. Na przykład **rdoEnviroment** odpowiada obiektowi **Workspace**,

rdoConnection odpowiada obiektowi **Database**, **rdoResultset** − obiektowi **Recordset** [Pelikant 2009].



Rys. 6. Model RDO

Źródło: opracowanie własne na podstawie [Pelikant 2009].

Wśród zalet RDO wyróżnić można możliwość uzyskania dostępu do źródeł danych ODBC bez użycia lokalnego procesora zapytań, co skutkuje większą wydajnością i elastycznością dostępu do zdalnych silników baz danych. Wśród wad RDO wymienia się zaś spowolnienie pracy aplikacji oraz większe zapotrzebowanie na pamięć.

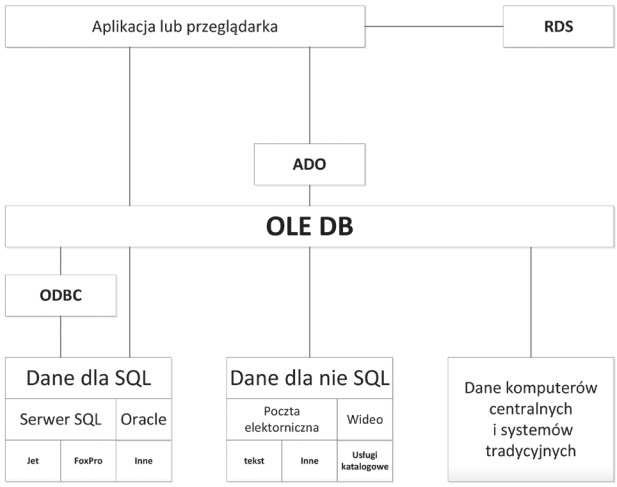
2.4. OLE DB – *Object Linking and Embedding for DataBase*

Interfejs ten stanowi rozwinięcie opracowane przez firmę Microsoft standardu ODBC. Głównym założeniem OLE DB jest uogólnienie modelu klient-serwer ODBC do modelu konsument danych-dostawca danych. Dostawca danych rozumiany jest zazwyczaj jako program dostarczający dane innemu programowi, konsument danych jest to zaś program będący biorcą danych użytkowych lub też w formie metadanych [Freeze 2001].

W OLE DB dostawcą danych może być każdy obiekt generujący dane (np. arkusz kalkulacyjny, pliki tekstowe − odpowiednio separowane, inne aplikacje) w przeciwieństwie do ODBC. OLE DB umożliwia aplikacjom współdzielenie i manipulację zbiorami danych w taki sposób, jak obiektami. Zapewnia niskopoziomowy dostęp do każdego źródła danych, także do relacyjnych i nierelacyjnych baz danych, systemów plików, poczty elektronicznej, tekstów czy grafiki [Connolly, Begg 2004].

Co ważne, jest to duży i dość złożony interfejs na poziomie systemu, a nie aplikacji. Bezpośredni dostęp możliwy jest dla programistów biegłych w języku C. Tworzone aplikacje mogą uzyskiwać bezpośredni dostęp do danych za pomocą OLE DB bądź też za pośrednictwem OLE DB mogą wywoływać ODBC, w celu uzyskania dostępu do źródeł danych ODBC, co zilustrowano na rys. 7.

Wśród zalet OLE DB wyróżnić można dostęp do różnych źródeł danych, w tym również nierelacyjnych, do wad OLE DB zalicza się zaś fakt, iż jest to technologia dedykowana tylko dla rozwiązań w środowisku MS Windows, możność operowania



Rys. 7. Architektura OLE DB

Źródło: opracowanie własne na podstawie [Connolly, Begg 2004].

na danych wymaga implementacji zbioru interfejsów składających się na OLE DB zarówno przez odbiorców, jak i dostawców danych, co jest trudne w bezpośredniej realizacji i wymaga ogromnej wiedzy i doświadczenia programistycznego.

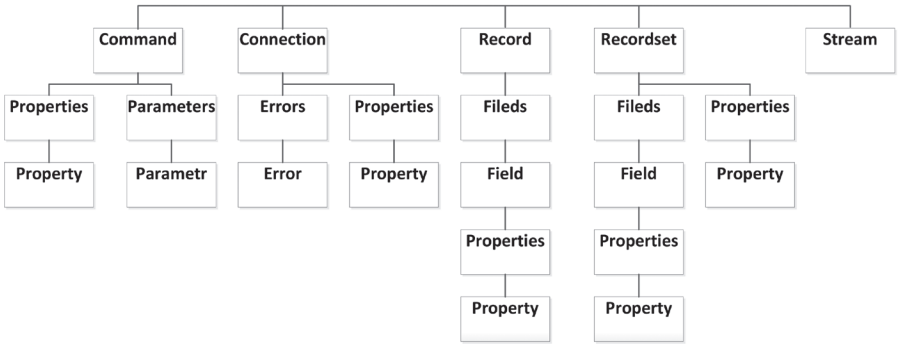
2.5. ADO – *ActiveX Data Objects*

Opracowany w 1996 r., stanowi obiektową nakładkę na OLE DB, pozwalającą względnie łatwo tworzyć aplikacje korzystające z OLE DB. Jest warstwą odnoszącą się do interfejsu na poziomie aplikacji, a nie systemu operacyjnego, tym samym jest technologią przeznaczoną dla programistów. Za pośrednictwem OLE DB umożliwia dostęp nie tylko do baz danych, lecz również do innych struktur, jak pliki tekstowe, pliki arkusza kalkulacyjnego czy pliki html. ADO umożliwiło programistom Visual Basica dostęp do niskopoziomowych aspektów paradygmatu OLE DB. Ze względu na to, iż jest to interfejs opracowany przez Microsoft, siłą rzeczy wręcz doskonale współpracuje z bazami danych MS Access czy MS SQL Server. W przypadku serwerów baz danych innych producentów warto wziąć pod uwagę sterowniki innych dostawców OLE DB, co powinno poprawić wydajność i efektywność dostępu do danych [ADO; Wybrańczyk 2004].

ADO w całości jest oparte na warstwie połączeniowej. Zawiera wszystkie niezbędne mechanizmy pozwalające nawigować po zbiorach i obsługiwać transakcje. Model ADO również ma hierarchiczną strukturę, co przedstawiono na rys. 8 [Roman 2001].

Model ADO zawiera trzy podstawowe obiekty, pozwalające uzyskać dostęp do danych: **Command** − przechowuje niezbędne instrukcje SQL do wykonania lub procedury przechowywanej, **Connection** − wskazuje ścieżkę źródła danych,

Recordset − przechowuje wynik działania zapytania. Początkiem każdego działania może być każdy z tych trzech obiektów w przeciwieństwie do DAO, w którym taką rolę odgrywał obiekt DBEngine [Pelikant 2009].



Rys. 8. Model ADO

Źródło: opracowanie własne na podstawie [Pelikant 2009].

Na dostawcy danych spoczywa decyzja, jakie elementy modelu obiektowego ADO zostaną zaimplementowane i w jakim zakresie.

Wśród zalet ADO wymienia się możliwość nawiązania bezpośredniego połączenia bez konieczności definiowania plików wymiany (wymagane są tylko odpowiednie sterowniki), łatwość użycia technologii, znacznie większą szybkość działania w stosunku do poprzednich technologii, niewielkie zużycie pamięci oraz przestrzeni dyskowej. Wśród wad ADO wyróżnić można niejednolitą i złożoną składnię łańcucha połączeń w zależności od użytego dostawcy, dostępność tylko dla platformy Windows − brak możliwości zastosowania w aplikacjach międzyplatformowych.

2.6. RDS – *Remote Data Services*

W zasadzie jest to część interfejsu ADO, która jest w stanie obsługiwać zdalne rekordy. Technologię stworzono na potrzeby tworzenia skryptów do serwera MS IIS (*Internet Information Server*). Pozwala ona manipulować bazą danych po stronie klienta za pośrednictwem Internetu [RDS].

RDS wykorzystuje technologię ADO po stronie serwera do wykonywania zapytań oraz przesyłania wyników, w postaci zestawu rekordów do klienta. Przy tym klient ma możliwość uruchamiania dodatkowych zapytań na zwróconym zestawie danych. Zawiera mechanizmy pozwalające przesłać zaktualizowane dane do serwera WWW oraz mechanizm buforowania, dzięki któremu zwiększa się wydajność aplikacji przez redukcję liczby odwołań do serwera WWW. W przeciwieństwie do klasycznego ADO, które utrzymuje połączenie z bazą danych, działa zawsze na lokalnym, zwróconym zbiorze rekordów [Connolly, Begg 2004].

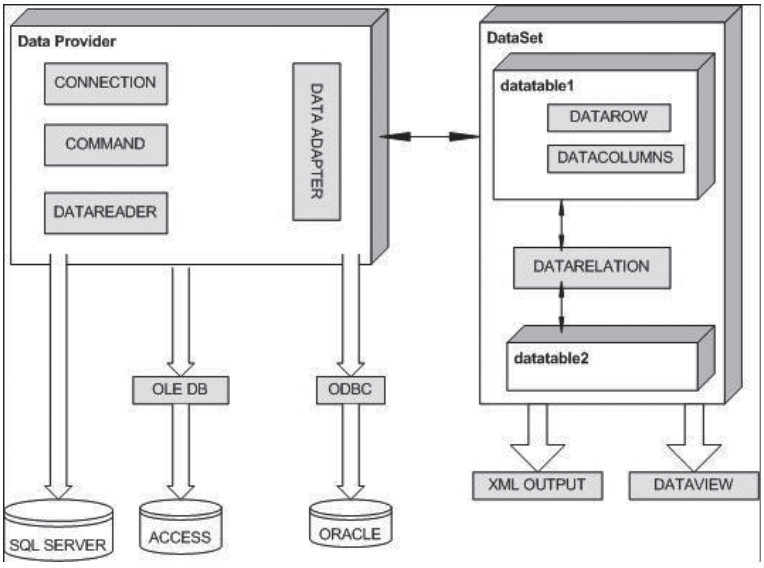
Wśród zalet RDS wyróżnić można dostęp do bazy danych za pośrednictwem Internetu i brak wymagań stałego połączenia z bazą. Do jej wad zalicza się fakt, że jest to technologia skojarzona z ADO, dostępna na platformie Windows.

2.7. ADO.NET − *Active Data Objects.NET*

Po raz pierwszy technologia pojawiła się w 2002 r. W przeciwieństwie do klasycznego modelu ADO, który korzystał z obiektów COM (*Component Object Model*) obsługiwanych tylko w środowisku Windows, w ADO.NET firma Microsoft wprowadziła obsługę XML-a. Dzięki temu w założeniach ADO.NET miało działać w zasadzie na każdej platformie, która obsługuje standard XML. Jest to technologia ukierunkowana głównie na tworzenie aplikacji wielowarstwowych, rozproszonych na kilku serwerach [ADO.NET1].

W odniesieniu do klasycznego ADO ADO.NET korzysta z dwóch warstw: połączeniowej i bezpołączeniowej. Warstwa połączeniowa jest mocno powiązana z używanym dostawcą danych, najczęściej z konkretnym SZBD. Warstwa bezpołączeniowa jest niezależna od dostawców danych. Wykorzystywana jest, gdy nie jest wymagany ciągły dostęp do źródła danych. Oferuje możliwość pracy na danych tymczasowych, które mogą być odczytywane i pobierane w strumieniach xml, jak i plikach. Dane pobrane ze źródła danych przechowywane i przetwarzane są na lokalnym komputerze użytkownika, zazwyczaj w postaci lokalnej bazy danych [ADO.NET2].

Na rysunku 9 zaprezentowano architekturę ADO.NET.



Rys. 9. Architektura ADO.NET

Źródło: [http://www.codeproject.com/Articles/28819/ADO-NET-Interview-Questions-Part-1].

Model ADO.NET wprowadza pojęcie przestrzeni nazw, które pełnią funkcję kontenerów dla wielu istotnych klas wykorzystywanych do gromadzenia i zarządzania danymi. Przestrzeń **System.Data** jest kontenerem dla klas warstwy bezpołączeniowej: **DataSet** (przechowuje odłączone i zbuforowane dane), **DataTable** (przechowuje zawartość tabel), **DataRow** (przechowuje dane, obiekt tej klasy reprezentuje pojedynczy wiersz DataTable), **DataColumns** (przechowuje informacje o strukturze tabeli, obiekt tej klasy reprezentuje schemat pojedynczej kolumny w klasie DataRow) i **DataRelation** (ilustracja związku między dwoma obiektami DataTable). Dzięki tym klasom możliwe jest uzyskanie dostępu do danych relacyjnych, ich przetwarzanie i przechowywanie. Dostęp do danych uzyskiwany jest za pośrednictwem następujących przestrzeni: **System.Data.OLEDB**, **SystemData.SQLClient**, **System.Data.ODBC**, **Sytem.Data.SQLServerCE**, **System.Data.ORACLEClient**, **System.Data.EntityClient**. Przestrzeń **System.XML** zawiera wszystkie niezbędne obiekty realizujące odczyt, zapis, przechowywanie i modyfikację dokumentów XML.

Przestrzeń **System.Data.Common** zawiera klasy warstwy połączeniowej, współużytkowane przez dostawców danych. Dostawca danych zapewnia dostęp do źródeł danych w postaci binarnej biblioteki. Dostawca danych .NET składa się z kilku klas: **Connection** (pozwala na nawiązanie i zamykanie połączenia ze źródłem danych), **DataAdapter** (odgrywa rolę łącznika pomiędzy warstwą połączeniową i bezpołączeniową; wprowadza lokalne dane do obiektów DataSet i DataTable, jak również je uaktualnia na podstawie zawartości źródła danych), **Command** (pozwala na modyfikację danych, wykonywanie instrukcji DDL, zapytań czy też procedur), **DataReader** (służy do odczytu) [Thomsen 2004].

Wśród zalet ADO.NET wyróżnia się następujące cechy:

- Dla warstwy połączeniowej: definiowanie połączenia ze źródłem danych przeprowadzane jest tylko jednokrotnie, aplikacja uzyskuje dostęp do mechanizmów bezpieczeństwa oferowanych przez konkretne SZBD, z którymi nawiązano połączenie.
- Dla warstwy bezpołączeniowej: mniejsze zużycie zasobów serwera (zużywane są tylko w momencie zapotrzebowania), wygodne podejście dla aplikacji wielowarstwowych, jak również WEB-owych, wydajniejsza modyfikacja danych (ze względu na czas dostępu).

Wśród wad ADO.NET wyróżnić zaś można:

- Dla warstwy połączeniowej: tryb nieużyteczny z punktu widzenia architektury wielowarstwowej aplikacji, zwiększone zużycie zasobów serwera, niewielka skalowalność aplikacji, nieprzystosowanie dla aplikacji opartych o technologie WEB-owe.
- Dla warstwy bezpołączeniowej: zużywanie zasobów komputera klienckiego, uboższy wachlarz dostępnych opcji związanych z bezpieczeństwem danych, zwiększenie kosztów związanych z wielokrotnym otwieraniem i zamykaniem połączenia ze źródłem danych, jak również kosztów związanych z czasem pobierania dużych porcji danych z serwera.

3. Zestawienie wybranych cech interfejsów

W punkcie tym w ujęciu tabelarycznym zaprezentowano zestawienie wybranych cech przedstawionych w artykule interfejsów dostępu do baz danych.

Uwzględnione zostały zestawienia nt. obsługiwanych systemów operacyjnych, daty opracowania i producenta interfejsu, niezależności od systemu operacyjnego, języka programowania i bazy danych, obsługi baz SQL, obsługi baz nierelacyjnych, obsługi baz plikowych i plików.

Tabela 1. Wykaz obsługiwanych platform systemów operacyjnych

Lp.	Interfejs	Win32	Win64	OS X	Linux	iOs	Android
1	ODBC	✓	✓	✓	✓	✓	✓
2	DAO	✓	–	–	–	–	–
3	RDO	✓	–	–	–	–	–
4	OLE DB	✓	✓	–	–	–	–
5	ADO	✓	–	–	–	–	–
6	RDS	✓	–	–	–	–	–
7	ADO.NET	✓	✓	–	–	–	–

Źródło: opracowanie własne.

Tabela 2. Wykaz lat powstania interfejsów

Lp.	Interfejs	Rok	Producent
1	ODBC	09.1992	Microsoft
2	DAO	11.1992	
3	RDO	1995	
4	OLE DB	08.1996	
5	ADO	10.1996	
6	RDS	1997	
7	ADO.NET	2002	

Źródło: opracowanie własne.

Tabela 3. Zestawienie nt. niezależności od systemu operacyjnego, języka programowania i bazy danych

Lp.	Interfejs	Niezależność od systemu operacyjnego	Niezależność od języka programowania	Niezależność od bazy danych
1	2	3	4	5
1	ODBC	✓	✓	✓
2	DAO	–	–	✓

Tabela 3, cd.

1	2	3	4	5
3	RDO	–	–	✓
4	OLE DB	–	–	✓
5	ADO	–	✓	✓
6	RDS	–	✓	✓
7	ADO.NET	–	✓	✓

Źródło: opracowanie własne.

Tabela 4. Zestawienie nt. obsługi baz SQL, obsługi baz nierelacyjnych, obsługi baz plikowych i plików

Lp.	Interfejs	Obsługa plików	Obsługa baz plikowych	Obsługa baz SQL	Obsługa baz nierelacyjnych
1	ODBC	✓	✓	✓	–
2	DAO	–	✓	✓	–
3	RDO	–	✓	✓	–
4	OLE DB	✓	✓	✓	✓
5	ADO	✓	✓	✓	✓
6	RDS	✓	✓	✓	✓
7	ADO.NET	✓	✓	✓	✓

Źródło: opracowanie własne.

4. Zakończenie

Microsoft w ciągu zaledwie jednej dekady opracował dość pokaźną liczbę interfejsów umożliwiających nawiązanie łączności aplikacji ze źródłami danych. Poszczególne technologie różnią się uniwersalnością rozwiązań, mając na uwadze niezależność od platformy systemowej, jak np. ODBC (dostępne są komercyjne sterowniki ODBC dla popularnych serwerów SQL dla Linuxa [EASYSOFT DRIVERS; ODBC Linux], dostarczane m.in. przez EasySoft [EASYSOFT], co więcej − oferują dostęp również do nierelacyjnych baz danych, jak MongoDB, sterowniki ODBC dla Mac OS X dostarczane przez Actual Technology [ODBC Mac OS X], EasySoft, sterowniki ODBC SDK dla iOS dostarczane przez ODBC Router [ODBC SDK iOS]), pozostałe zaś są ściśle powiązane z system operacyjnym Windows. Część technologii, jak DAO, RDO jest w zasadzie zależna od konkretnego języka programowania (głównie dostępna z poziomu języka Visual Basic). Pozostałe są osiągalne z poziomu języków wchodzących w skład pakietu Visual Studio. Praktycznie wszystkie zaprezentowane w artykule interfejsy cechuje niezależność od konkretnej

platformy bazodanowej. Pewne zastrzeżenia w tej kwestii może budzić technologia DAO. Wiążą się z tym określone utrudnienia związane z koniecznością definiowania plików wymiany DSN.

Nie wszystkie rozwiązania radzą sobie także z dostępem do nierelacyjnych źródeł danych. Wyjątek stanowi OLE DB i ADO.NET, które daje możliwość dostępu do hierarchicznych baz danych, takich jak serwer Exchange czy choćby Active Directory.

Jedne z opracowanych przez Microsoft rozwiązań stanowią rozwinięcie wcześniejszych technologii. Przykładem jest stworzenie RDO (na bazie wcześniejszej technologii DAO), które dość szybko zostało wyparte przez model ADO. Pewne rozwiązania z ADO zostały przeniesione do ADO.NET, choć samo ADO.NET cechuje się na tyle odmiennym podejściem, iż trudno pokusić się o opinię, że bezpośrednio wywodzi się z samego ADO. Drugą grupę stanowią interfejsy o innym podejściu do realizacji dostępu do danych. Do tej grupy należą ODBC i OLE DB. Stanowią one też grupę niskopoziomowych mechanizmów wymiany danych, podczas gdy DAO, RDO, ADO należą do grupy interfejsów będących nakładkami programowymi na niskopoziomowy mechanizm (odpowiednio ODBC i OLE DB).

Jedne z mechanizmów dostępu do danych omawianych w tym artykule są nadal rozwijane, jak choćby ADO.NET (która stała się wiodącą technologią dostępu do baz danych promowaną przez Microsoft), inne już nie. Wraz z upływem czasu zatrzymany został rozwój technologii DAO, RDO, ADO.

Aktualnie najczęściej używaną i zarazem wiodącą technologią Microsoft jest ADO.NET. Interfejsy ODBC i OLE DB nie są już tak popularne jak kiedyś (m.in. ze względu na efektywność dostępu czy też brak dostępu do specjalizowanych funkcji najnowszych wersji SQL Servera − nie wszystkie zostały uwzględnione w sterownikach OLE DB, natomiast zostały zawarte w nowym silniku Native Client), ale nadal znajdują swoje zastosowanie, choć w znacznie mniejszym stopniu niż przed laty.

Oczywiście, nie są to jedyne opracowane interfejsy dostępowe do baz danych. Gdy Microsoft opracowywał swoje własne rozwiązania, w międzyczasie powstało wiele innych technologii opracowanych przez innych producentów. Wystarczy wymienić takie firmy, jak Borland, Sun czy Embarcadero. Niektóre z rozwiązań oferowanych przez konkurencję Microsoftu w pewnym – niewielkim, stopniu bazują na jego osiągnięciach, z kolei inne oparte są na zupełnie odmiennej koncepcji. Konkurencyjne rozwiązania względem tych oferowanych przez Microsoft zostały zaprezentowane w kolejnym artykule, stanowiącym drugą część cyklu poświęconego zagadnieniom realizacji dostępu do baz danych [Łacheciński 2015].

Literatura

ADO, https://msdn.microsoft.com/en-us/library/ms675532(v=vs.85).aspx (10.10.2015).
ADO ODBC OLE DB, https://support.microsoft.com/en-us/kb/190463 (12.10.2015).
ADO.NET1, https://msdn.microsoft.com/pl-pl/library/h43ks021(v=vs.110).aspx (15.10.2015).

ADO.NET2, http://www.codeproject.com/Articles/28819/ADO-NET-Interview-Questions-Part-1.
Barczak A., Zacharczuk D., *Analiza porównawcza technologii dostępu do baz danych w środowisku Borland Delphi*, II Krajowa Konferencja Naukowa Technologie przetwarzania danych, 24-26.09.2007, Poznań, http://www.cs.put.poznan.pl/kkntpd/tpd_pliki/publikacja/.
Connolly T., Begg C., 2004, *Systemy baz danych. Praktyczne metody projektowania, implementacji i zarządzania,* tom 2, RM, Warszawa.
DAO, http://www.fmsinc.com/TPapers/dao/index.html (10.10.2015).
Darakhvelidze P., Markov E., 2005, *Delphi. Techniki bazodanowe i internetowe*, Helion, Gliwice.
EASYSOFT DRIVERS, http://www.easysoft.com/products/data_access/index.html#odbc-drivers (22.10.2015).
EASYSOFT, http://www.easysoft.com/ (15.10.2015).
Freeze W., 2001, *Visual Basic 6. Programowanie baz danych*, Helion, Gliwice.
http://www.codeproject.com/Articles/28819/ADO-NET-Interview-Questions-Part-1.
Łacheciński S., 2015, *Interfejsy dostępu do baz danych − przegląd technologii Borland, Embarcadero, Sun, Oracle*, Informatyka Ekonomiczna 3 (37) 2015, Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu, Wrocław.
Mościcki A., 2006, *Ewolucja i efektywność technologii dostępu do baz danych*, Software Developer's Journal, 11/2006, s. 34-43.
ODBC Linux, http://www.easysoft.com/developer/interfaces/odbc/linux.html (22.10.2015).
ODBC Mac OS X, http://www.actualtech.com/products.php (22.10.2015).
ODBC SDK iOS, http://odbcrouter.com/ipad (22.10.2015).
Pelikant A., 2009, *Bazy danych. Pierwsze starcie*, Helion, Gliwice.
RDO, https://msdn.microsoft.com/en-us/library/aa733681(v=vs.60).aspx (10.10.2015).
RDS, https://msdn.microsoft.com/en-us/library/ms676188(v=vs.85).aspx (10.10.2015).
Roman S., 2001, *Access. Baza danych − projektowanie i programowanie*, Helion, Gliwice.
Thomsen C., 2004, *Programowanie baz danych w Visual Basic.NET*, Mikom, Warszawa
Wybrańczyk M., 2004, *Delphi 7 i bazy danych*, Helion, Gliwice.