ISSN 1507-3858
e-ISSN 2450-0003

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

INTERFEJSY DOSTĘPU DO BAZ DANYCH – PRZEGLĄD TECHNOLOGII BORLAND, EMBARCADERO, SUN, ORACLE

INTERFACES OF ACCESS TO DATABASES – REVIEW OF BORLAND, EMBARCADERO, SUN, ORACLE TECHNOLOGIES

DOI: 10.15611/ie.2015.3.04
JEL Classification: C88, L63, L86

Streszczenie: Artykuł poświęcono niezwykle istotnemu zagadnieniu, jakim jest realizacja dostępu aplikacji do różnych źródeł danych. Przedstawia on rozwój, jaki miał miejsce na przestrzeni ostatniego ćwierćwiecza w odniesieniu do najbardziej popularnych interfejsów dostępu do danych. Dzięki nim realizowany jest dostęp aplikacji do danych przechowywanych i zarządzanych przez różne serwery SQL. Zaprezentowana została architektura wybranych interfejsów opracowanych i rozwijanych przez takich potentatów z branży IT jak: Borland, Embarcadero, Sun czy Oracle. Celem artykułu jest wyznaczenie użyteczności analizowanych interfejsów z jednoczesnym wskazaniem wad i zalet poszczególnych technologii. Artykuł jest analizą porównawczą prezentowanych w nim interfejsów.

Słowa kluczowe: interfejs dostępu do baz danych, BDE, BDP.NET, DBX, IBX, FireDAC, JDBC, JDO.

Summary: This article is devoted to an extremely important issue, which is to implement applications access to various data sources. It shows the development that took place during the last quarter of the 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 Borland, Embarcadero, Sun, Oracle. 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 database, BDE, BDP.NET, DBX, IBX, FireDAC, JDBC, JDO.

1. Wstęp

W artykule opisano interfejsy opracowane przez firmę Borland, Embarcadero, Sun, Oracle, a także technologie opracowane przez inne podmioty, jak choćby produkty firmy Devart, dedykowane konkretnym serwerom SQL.

Z upływem lat część z tych technologii została stopniowo wygaszona, a pozostawiona głównie ze względu na zachowanie kompatybilności starszych wersji oprogramowania stworzonych z ich użyciem (m.in. w celu pozostawienia możliwości migracji do nowszych rozwiązań). Niektóre zaś rozwiązania nadal są rozwijane pod skrzydłami innych firm, tak jak ma to miejsce w przypadku interfejsu Borlanda dbExpress, przejętego przez giganta rozwiązań bazodanowych Embarcadero, czy też JDBC opracowanego przez firmę Sun, a od kilku lat przejętego i rozwijanego przez firmę Oracle. W międzyczasie pojawiły się także całkiem nowatorskie podejścia, jak choćby rozwiązanie FireDAC firmy Embarcadero.

2. Wybrane interfejsy

W punkcie tym prezentowane są interfejsy dostępu do baz danych dostępne z poziomu języków programowania: Delphi, C++ Builder, Lazarus, Java.

2.1. BDE – Borland DataBase Engine

Realizuje założenia standardu IDAPI (Independent Database Application Programming Interface) opracowanego na początku lat 90. XX wieku przy współpracy kilku firm: Borland, Microsoft, Hewllet Packard, Sun, IBM, ORACLE. Stanowił alternatywę dla ODBC, jednocześnie będąc technologią bardziej wydajną od ODBC. BDE łączyło zalety ODBC i IDAPI, wprowadzając nowe rozwiązanie w postaci bezpośrednich sterowników dla najpopularniejszych, jak na tamte czasy, serwerów bazodanowych, które nazwano SQL Links. Zarówno ODBC, jak i IDAPI pozwalały na użycie plikowych baz danych, natomiast nowo wprowadzone mechanizmy w BDE pozwoliły z nich korzystać, jak z serwerów SQL, co znacznie ułatwiło migrację rozwiązań opartych na plikach do serwerów SQL. Technologia umożliwiała tworzenie aplikacji w architekturze klient-serwer [Barczak, Zacharczuk; BDE].

Należy podkreślić, iż to właśnie w BDE po raz pierwszy pojawiło się pojęcie silnika bazodanowego. Był to pierwszy, stworzony na tak ogromną skalę uniwersalny silnik bazodanowy, dostępny dla programistów ze środowiska Delphi [Mościcki, Kruk 2006].

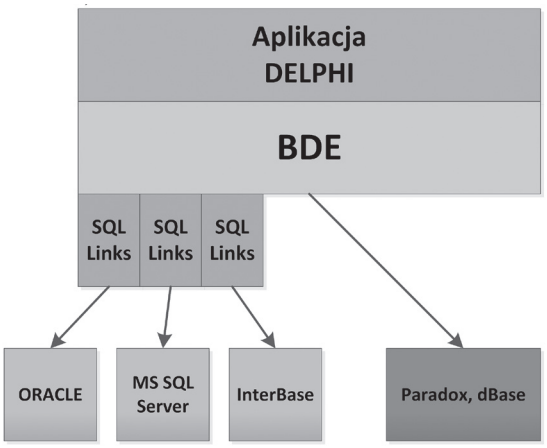
Również w BDE po raz pierwszy pojawił się własny bufor rekordów w postaci zbioru danych DataSet (w postaci struktury kolumn i wierszy, która stanowi warstwę pośrednią między komponentami obsługi danych a komponentami warstwy prezentacji danych).

Do prawidłowego działania aplikacji wymagana była instalacja na komputerach klientów BDE. Aplikacja klienta komunikuje się z lokalną kopią BDE. Specjalne

sterowniki SQL Links odpowiadały za translację poleceń BDE na specyficzne instrukcje dla konkretnego SZBD. Dzięki takiemu podejściu możliwe stało się łączenie aplikacji w zasadzie z dowolną bazą danych, bez znajomości jej szczegółów implementacyjnych [Mościcki 2006].

BDE umożliwia dostęp do wielu baz danych: dBase, Paradox, FoxPro, Access, SQL Server, Sybase, Interbase, ORACLE, DB2, Informix. Ponadto za pośrednictwem sterowników ODBC możliwa jest komunikacja z innymi bazami danych [Wybrańczyk 2004].

Mimo wielu swoich zalet, jak na tamte czasy, firma Borland (mając na uwadze wysokie koszty dalszego rozwijania BDE) postawiła na nowy standard, bardziej elastyczny, tym samym spowalniając rozwój technologii BDE, a w 2002 r. definitywnie zaprzestano ją rozwijać. W kolejnych wersjach środowiska Delphi i C++ Builder umieszczano BDE w celu uzyskania kompatybilności wstecznej. Ostatnią wersją, do której dołączono BDE, była XE6, z 15 kwietnia 2014 r. (choć nadal jest możliwość jego doinstalowania).



Rys. 1. Architektura BDE

Źródło: opracowanie własne.

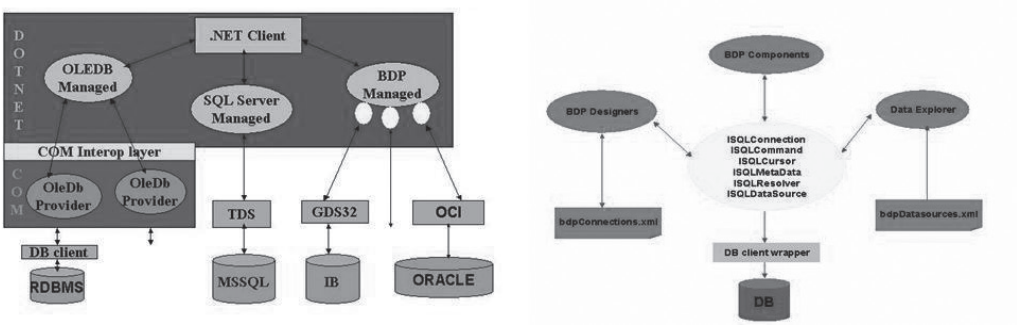
Wśród zalet BDE wymienia się: dostęp do wielu popularnych serwerów SQL za pośrednictwem bezpośrednich sterowników, co wówczas znacznie wyprzedzało konkurencję, łatwiejszą migrację rozwiązań plikowych do serwerów SQL, możliwość nawiązania połączenia praktycznie z dowolną bazą bez znajomości jej szczegółów implementacyjnych. Wśród wad BDE wyróżnić zaś można konieczność instalacji klienta BDE na stacjach roboczych i fakt, że jest to technologia przystosowana do realizacji raczej niewielkich, lokalnych projektów.

2.2. BDP.NET – Borland Data Provider for .NET

Standard wprowadzony w 2003 r. w odpowiedzi na opracowanie środowiska .NET przez firmę Microsoft. Po raz pierwszy dostępny był w wersji Delphi 8. Mimo że BDP.NET należy do kodu zarządzalnego w środowisku .NET, jest produktem firmy Borland. Wymaga jednakże dołączenia dodatkowych podzespołów do systemu, na którym będzie uruchamiana aplikacja. Wynika to z faktu, iż Borland udostępnił dodatkowe mechanizmy, które rozszerzały wachlarz dostępnych serwerów (względem klasycznego ADO.NET), z którymi aplikacja mogła nawiązać połączenie, m.in. o serwer InterBase czy IBM DB2.

Samo ADO.NET zapewniało obsługę serwerów: MS SQL Server, ORACLE oraz technologii OLEDB. Technologia BDP.NET była rozwijana i utrzymywana przez firmę Borland do 2007 r. Ostatnią wersją środowiska, do którego była dołączona, była wersja Delphi 2007, wypuszczona na rynek pod egidą firmy Codegear.

Na rysunku 2 zaprezentowano architekturę BDP.NET [BDP.NET].



Rys. 2. Architektura BDP.NET

Źródło: [http://edn.embarcadero.com/article/31939].

Wśród zalet BDP.NET wyróżnić można szybkie tworzenie wszechstronnych i niezawodnych aplikacji na podstawie możliwości oferowanych przez platformę. NET oraz środowisko programistyczne Delphi, aplikacji WWW w oparciu na ASP. NET i XML Web Services, aplikacji bazodanowych w oparciu na ADO.NET i BDP. NET, tworzenie aplikacji na bazie WinForms i VCL Controls. Wśród wad BDP.NET wymienia się zaś konieczność instalacji dodatkowych podzespołów na stacjach roboczych i fakt, że jest to technologia nierozwijana od 2007 r.

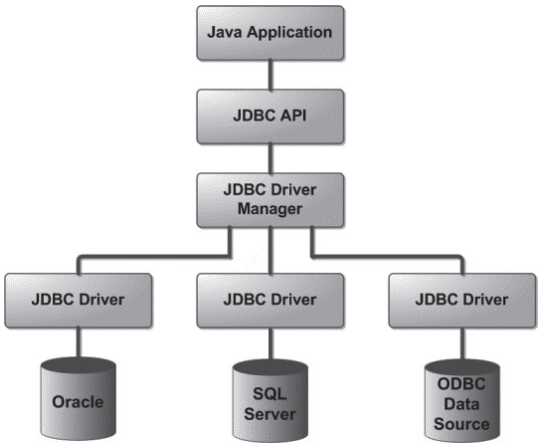
2.3. JDBC – Java DataBase Connectivity

Interfejs stworzyła firma Sun Microsystem w 1996 r. Opracowany został na wzór specyfikacji ODBC z uwzględnieniem odmiennej specyfikacji Javy. Ma także nieco większe możliwości niż ODBC. W styczniu 2010 r. Sun Microsytem przejęła firma

Oracle i od tej pory i Java, i JDBC rozwijane są w ramach Oracle. Jednocześnie JDBC, SQLJ stanowią podstawową technologię dostępu do serwerów ORACLE oferowanych przez firmę Oracle [SQLJ, JDBC; Wojciechowski, Zakrzewicz 2002].

Aplikacje tworzone za pomocą języka Java uzyskują niezależny od platformy sprzętowej, bazodanowej i systemu operacyjnego dostęp do baz danych. Niezależność od systemu operacyjnego oferuje Java jako język przenośny, niezależny od architektury i platformy [JDBC DevelopersBook; JDBC docs; JDBC Stanford].

Zapytania SQL przesyłane są do modułu sterującego SZBD bez sprawdzenia zgodności i możliwości wykonania. Gdy zapytanie nie może być poprawnie obsłużone przez SZBD, generowany jest wyjątek. Rozbudowane aplikacje, w celu sprawdzenia zgodności zapytania SQL, dostosowują się do wymagań konkretnego SZBD na podstawie metadanych zawierających m.in. dane o wymaganiach, ustawieniach danego SZBD [JDBC technetwork; JDBC technetwork overview].

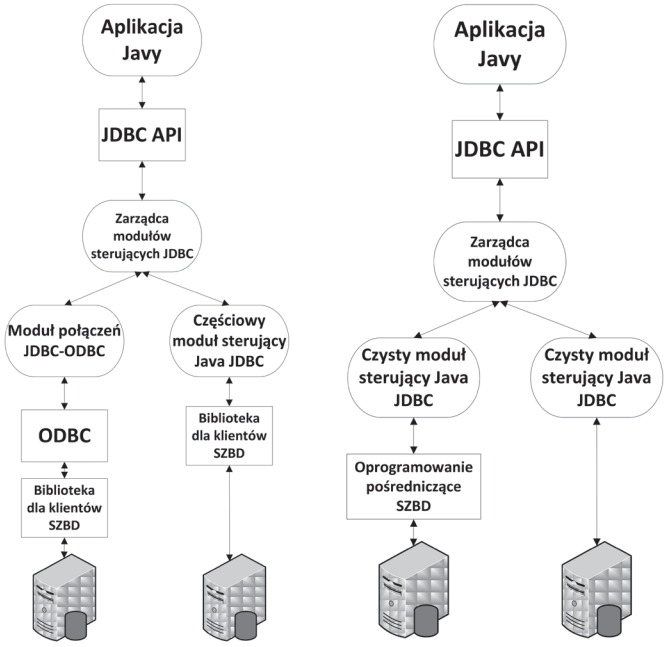


Rys. 3. Architektura JDBC

Źródło: [http://www.developersbook.com/jdbc/interview-questions/jdbc-interview-questions-faqs.php].

Standard JDBC pozwala tworzyć interfejsy programowe API wyższego poziomu: SQLJ − osadzony SQL dla Javy, oraz JavaBlend − bezpośrednie odwzorowanie tabel relacyjnej bazy danych na klasy Javy. Sama aplikacja dostęp do danych uzyskać może na kilka sposobów; są one następujące: czysty moduł JDBC z bezpośrednim połączeniem z bazą danych, czysty moduł sterujący JDBC dla oprogramowania pośredniczącego bazy danych, częściowy moduł sterujący JDBC, połączenie JDBC-ODBC.

W połączeniu JDBC-ODBC ODBC stanowi warstwę pośrednią między modułem sterującym JDBC a biblioteką dla aplikacji klienta dostarczaną przez producenta bazy danych. Kod binarny ODBC musi być załadowany na każdym komputerze, który używa modułu sterującego JDBC. Niestety, wpływa to na spore zmniejszenie użyteczności tego typu rozwiązania w Internecie. Ponadto wykonywana jest dodatkowa translacja między JDBC a ODBC.



Rys. 4. Architektura JDBC z wyszczególnieniem różnych wariantów połączeń

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

W rozwiązaniu z częściowym modułem sterującym wywołania JDBC zamieniane są na wywołania API klienta konkretnego SZBD. Aby moduł sterujący mógł skomunikować się z bazą danych, wymagane jest zainstalowanie na każdym kliencie stosownego oprogramowania klienckiego bazy danych. Rozwiązanie to ma nieco większą wydajność, tym samym całkiem nieźle nadaje się do wykorzystania w rozwiązaniach intranetowych.

W rozwiązaniu z czystym modułem sterującym Java JDBC i warstwą oprogramowania pośredniczącego dla SZBD wywołania JDBC tłumaczone są na protokół oprogramowania pośredniczącego producenta bazy danych. Następnie protokół ten poddawany jest translacji przez serwer oprogramowania pośredniczącego na protokół SZBD. Oprogramowanie pośredniczące odpowiada za łączność z wieloma różnymi bazami danych. Jest to rozwiązanie najbardziej elastyczne, pozwalające nawiązać łączność z wieloma różnymi serwerami baz danych i heterogenicznymi bazami danych.

W rozwiązaniu z czystym modułem Java JDBC do bezpośredniego połączenia z bazą danych wywołania JDBC transformowane są na protokół sieciowy bezpośrednio używany przez SZBD (dzięki czemu możliwe jest odwołanie klienta bezpośrednio do serwera baz danych). Rozwiązanie to jak najbardziej nadaje się do zastosowań internetowych (dzięki temu, że moduły sterujące mogą być pobierane dynamicznie).

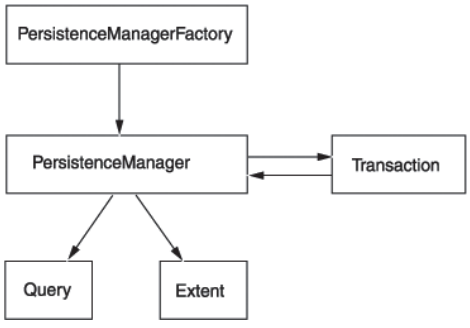
Moduły te są całkowicie zaimplementowane w Javie, dzięki czemu uzyskuje się niezależność od platformy oraz eliminacji ulegają także problemy instalacyjne (obecne w poprzednich rozwiązaniach). Jednakże wymagany jest inny moduł sterujący (dostarczany przez producentów baz danych) dla każdej bazy danych [Connolly, Begg 2004].

Wśród zalet JDBC wyróżnić można: uniwersalność zastosowania ze względu na platformy, prostotę składni, możliwość dynamicznego budowania treści poleceń SQL w trakcie działania aplikacji. Do wad JDBC można zaś zaliczyć: niezbyt dużą wydajność, zastosowanie w niedużych projektach, brak hermetyzacji, brak weryfikacji poprawności składni SQL na etapie kompilacji.

2.4. JDO – Java Data Objects

Pierwszą wersję interfejsu JDO 1.0 opracowano w 2002 r. w ramach Java Community Process JSR 12. Zmodernizowana wersja JDO 2.0 opublikowana została w 2006 r. W roku 2008 pojawiły się wersje JDO 2.1 i JDO 2.2 wydane przez Apache JDO. W 2010 r. opublikowana została wersja JDO 3.0.

JDO stanowi alternatywę dla JDBC. Udostępnia obiektowy mechanizm trwałości i standardowe interfejsy umożliwiające korzystanie z baz danych. Rekordy z relacyjnej bazy danych traktowane są jako obiekty. Współpracuje z relacyjnymi i obiektowymi bazami danych, jak również innymi rodzajami systemów. Na obiektowy model JDO składa się zestaw klas Javy i plik metadanych XML (zawierający wytyczne modelowania) [JDO; JDO docs; JDO Service architecture; JDO Service architecture; JDO technetwork; Tyagi i in. 2004].



Rys. 5. Model JDO

Źródło: [https://docs.oracle.com/cd/B14099_19/web.1012/b15901/sessions012.htm].

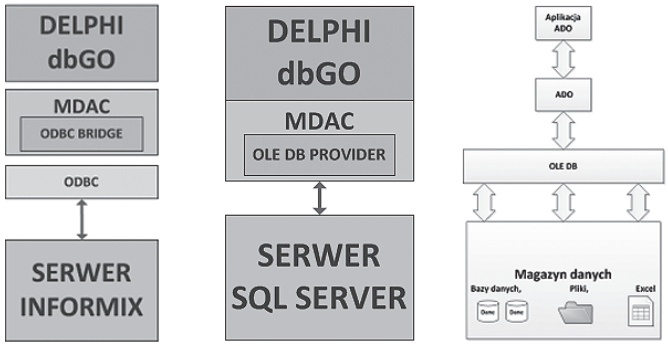
2.5. dbGo

Standard opracowany przez firmę Borland w odpowiedzi na wprowadzenie przez Microsoft interfejsu ADO. Początkowo technologia ta nosiła nazwę ADO Express, dopiero później została przemianowana na dbGo. Aby programiści Delphi i C++

Builder mogli korzystać z całego dobrodziejstwa, jakie na tamten czas oferowało ADO, w sposób podobny do BDE, wprowadzone zostały tzw. komponenty opakowujące możliwości ADO, bazując na klasycznym DataSet (w ADO używany był obiekt RecordSet) [Cantu 2003; Darakhvelidze, Markov 2005].

W odniesieniu do wcześniejszych technologii był on prostszy w użyciu, gdyż nie wymagał instalacji żadnych dodatkowych elementów na stacji roboczej. Ma silnik zbliżony do BDE. Interfejs dbGo wykorzystywany jest podobnie jak ADO dla aplikacji pracujących w środowisku Windows. Zawiera wszystkie niezbędne mechanizmy, takie jak: nawigacja po zbiorach czy realizacja transakcji [Wybrańczyk 2004].

Na rysunku 6 przedstawiono alternatywne sposoby dostępu do źródła danych za pomocą technologii dbGO.



Rys. 6. Architektura dbGo

Źródło: opracowanie własne na podstawie [Darakhvelidze, Markov 2005].

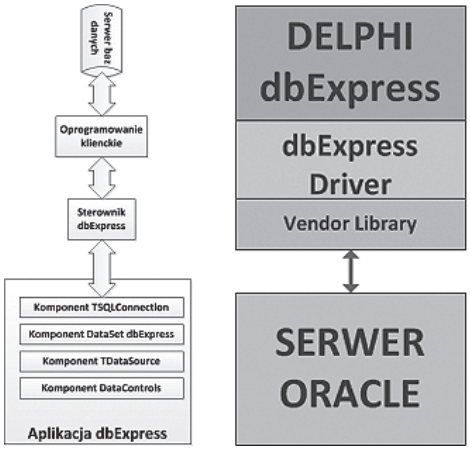
Wśród zalet dbGo wymienić można fakt, że jest to uniwersalna metoda dostępu do heterogenicznych źródeł danych, brak konieczności instalacji dodatkowych bibliotek. Wśród wad dbGo wyróżnić można, podobnie jak w odniesieniu do klasycznego ADO, złożoną składnię łańcucha połączeń i brak możliwości użycia w aplikacjach międzyplatformowych.

2.6. DBX− DBExpress

Standard ten został opracowany i wprowadzony na rynek w 2000 r. przez firmę Borland. Postanowiono rozdzielić warstwę transportową danych od warstwy przetwarzania. Dzięki temu DataSet transportowy stał się prosty, a zarazem szybki w działaniu. Potrafi on pobierać dowolne dane w identyczny sposób, niezależnie od tego, z jakiego źródła pochodzą. Bezproblemowa stała się również zmiana serwerów poprzez zmianę tylko samych danych połączeniowych, natomiast cały kod aplikacji pozostaje bez zmian [Cantu 2003; Darakhvelidze, Markov 2005; DBX; Wybrańczyk 2004].

Technologię tę cechuje lekkość i przenośność, realizowane m.in. przez implementację jednokierunkowych kursorów (przy minimalnym obciążeniu serwera i sieci) i brak własnej pamięci podręcznej. Nie wymaga także konfigurowania komputerów klienckich. Umożliwia dostęp tylko do serwerów SQL, nie ma dostępu do plików lokalnych. Brak własnej pamięci podręcznej niwelowany jest za pomocą tandemu dwóch obiektów: ClientDataSet i DataSetProvider. Połączenie tej pary obiektów wraz z technologią dbExpress oferuje ogromne możliwości i nieprzeciętną uniwersalność względem technologii realizującej własne zadania w zawsze ściśle określony sposób. Sterowniki dbExpress w maksymalnym stopniu użytkują natywne sterowniki, w postaci bibliotek dll dostarczanych przez producentów baz danych [BDE, dbGo, DBX; Mościcki, Kruk 2006].

Na rysunku 7 przedstawiono architekturę dbExpress oraz sposób dostępu do danych za pośrednictwem interfejsu dbExpress.



Rys. 7. Architektura dbExpress

Źródło: opracowanie własne na podstawie [Darakhvelidze, Markov 2005].

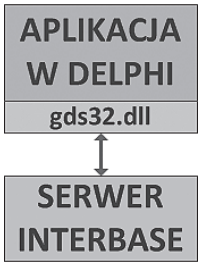
Wśród zalet dbExpress wyróżnić można: szybkość działania (komponenty komunikują się bezpośrednio z oprogramowaniem klienckim), dostęp do specjalnych funkcji serwera baz danych, przenośność będącą konsekwencją łatwego rozpowszechniania aplikacji (do każdej aplikacji wystarczy dołączyć odpowiedni sterownik, który jest automatycznie wykrywany przez dbExpress za każdym uruchomieniem aplikacji). Wśród wad dbExpress wyróżnić zaś można: użycie nieprzewijalnych kursorów, ograniczoną edycję danych (możliwa przez buforowanie po stronie klienta lub zastosowanie kwerend modyfikujących), fakt, że jest to technologia niezbyt użyteczna w bardzo zaawansowanych aplikacjach klient-serwer.

2.7. IBX – InterBase Express

IBX to technologia specjalnie opracowana dla serwera InterBase. Pozwala na specjalizowany dostęp do baz danych opartych o serwer InterBase. Dzięki temu możliwe jest wykorzystanie zaawansowanych możliwości oferowanych przez serwer InterBase, które nie są osiągalne za pośrednictwem innych rozwiązań ogólnego przeznaczenia [Cantu 2003; Wybrańczyk 2004].

Jest to technologia niezwykle wydajna, oferująca bardzo krótki czas łączenia z serwerem, jak również przetwarzania zapytań, co jest efektem wyeliminowania warstwy pośredniej między aplikacją a serwerem. Dodatkową zaletą jest to, iż cały mechanizm IBX (tak jak serwer InterBase) jest zaimplementowany w Delphi. Tak więc komponenty i klasy składające się na IBX są w pełni wkompilowane w plik wykonywalny aplikacji. Tworzona aplikacja nie wymaga dodatkowych konfiguracji, jak np. w przypadku BDE, ani też tworzenia złożonych instalatorów [IBX].

Oferuje ona zysk dużej efektywności i sterowania kosztem utraty elastyczności i przenośności rozwiązania w razie konieczności zmiany serwera baz danych [Mościcki 2006].



Rys. 8. Architektura InterBaseExpress

Źródło: opracowanie własne.

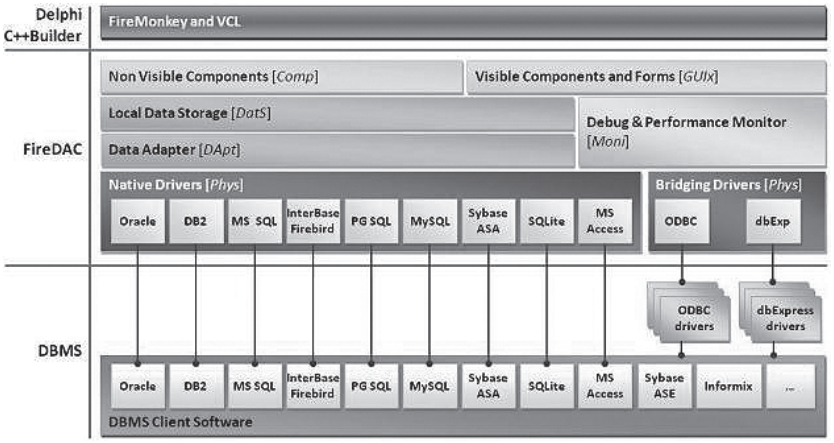
Wśród zalet IBX wyróżnić można: szybkość działania uzyskaną na skutek pominięcia warstwy pośredniej między aplikacją i serwerem oraz wykorzystania specjalnych funkcji serwera, możliwość użycia ponadstandardowych nowo implementowanych możliwości oferowanych przez serwer InterBase, uproszczoną dystrybucję aplikacji. Wśród wad IBX wyróżnić zaś można: niską skalowalność aplikacji w przypadku zaistnienia konieczności zmiany serwera bazodanowego, utratę elastyczności i uniwersalności aplikacji związaną ze zmianą serwera.

2.8. FireDAC

Punktem wyjścia do stworzenia interfejsu FireDAC było opracowanie w 2003 r. Components for Oracle. Następnie, w 2005 r., na podstawie wcześniejszych prac, udostępniony został pakiet FreeDAC 1.0. Pierwotnie technologia ta rozwijana była

pod nazwą AnyDAC pod skrzydłami firmy DA-SOFT Technologies, która była partnerem Embarcadero Technology, MySQL i FireBird Foundation Associate Member Supporting FireBird Development. Pierwsza wersja beta AnyDAC wprowadzona została w 2007 r., zaś w 2008 r. opublikowana została pełnoprawna wersja AnyDAC 2.0. Na przełomie lat 2012 i 2013 firma została wykupiona przez Embarcadero. Dzięki temu posunięciu produkt zyskał dodatkowe wsparcie i jest nadal prężnie rozwijany pod nazwą FireDAC. Początkowo AnyDAC był dołączany do najbardziej zaawansowanych wersji RAD Studio. Pierwszy FireDAC w pełni zintegrowany z RAD Studio pojawił się w wersji XE5.

FireDAC jest uniwersalną biblioteką dostępu do danych. Umożliwia tworzenie aplikacji, które komunikują się z wieloma korporacyjnymi bazami danych, a także działają na różnych urządzeniach (komputery, tablety, smartfony), pod kontrolą różnych systemów operacyjnych (Windows, Mac OS X, iOS, Android). Oferuje szybki i wydajny dostęp do baz danych różnych producentów. Zawdzięcza to uniwersalnej architekturze, która oprócz uniwersalnego dostępu do danych, korzysta także z natywnego dostępu do platform bazodanowych, tj.: Oracle, DB2, MS SQL Server, InterBase, Firebird, PostgreSQL, MySQL, Sybase, SQLite, MS Access, Informix, SQL Anywhere, AdvantageDB, DataSnap i wielu innych, z poziomu środowiska Delphi i C++Builder [FireDAC].



Rys. 9. Architektura FireDAC

Źródło: [http://www.embarcadero.com.pl/produkty/firedac/].

FireDAC stanowi niezwykle wydajną, a jednocześnie łatwą w użyciu warstwę dostępu do danych. Jednocześnie zapewnia pełną funkcjonalność wymaganą podczas tworzenia wydajnych aplikacji opartych na bazach danych. Spójny interfejs API FireDAC umożliwia obsługę wielu różnorodnych systemów baz danych, a co najważniejsze − zachowuje przy tym możliwość korzystania z unikalnych cech, specyficznych dla konkretnych serwerów baz danych.

Lekki, a zarazem bardzo elastyczny silnik może być używany bezpośrednio z poziomu aplikacji. Składa się z wielu komponentów bazodanowych i warstwy niewizualnej udostępnionych w postaci obiektowo zorientowanego API. Ponadto wyposażony jest w jeden z najszybszych zbiorów pamięciowych. Ma zaimplementowany również lokalny silnik SQL, pozwalający wykonywać zapytania SQL na pamięciowych zbiorach danych. Uniwersalny dostęp do danych pozwala tworzyć aplikacje, których kod jest dopasowywany do wielu różnych platform niezależnie od dialektu. Jednocześnie, dzięki wsparciu dla natywnych sterowników, uzyskiwany jest dostęp do zaawansowanych właściwości charakterystycznych, a zarazem unikatowych dla każdej bazy danych [FireDAC DocWiki].

Wśród zalet FireDAC wyróżnić można: uniwersalny sposób łączenia się z wieloma bazami danych, dostęp do zaawansowanych właściwości baz danych dzięki użyciu natywnych sterowników; dostępność dla wielu platform (Windows, Mac OS X, iOS, Android) i urządzeń, wysoko wydajny dostęp do danych. Wśród wad FireDAC wymienić zaś można dostępność tylko z poziomu środowiska Embarcadero RAD Studio.

2.9. Inne interfejsy

Na rynku istnieje również wielu innych dostawców interfejsów połączeniowych do baz danych. Wśród nich można wyróżnić m.in. firmę Devart czy też EasySoft. Firma Devart w oparciu o udostępnione źródło biblioteki dbExpress (opracowanej przez dawną firmę Borland) opracowała na przestrzeni lat wiele własnych, rozbudowanych interfejsów, za pomocą których można uzyskać specjalizowany dostęp do wybranych serwerów baz danych. Najczęściej są to technologie hybrydowe oferujące własny optymalizowany mechanizm dostępu bądź też w zależności od rodzaju serwera za pośrednictwem interfejsu OLE DB, bibliotek klienckich czy też bezpośredniego połączenia za pomocą protokołu TCP/IP. Funkcjonują również dostawcy oferujący specjalizowany dostęp do serwerów InterBase i FireBird [IBO; IBO Objects; IBO Codegear; FIB Plus].

Aplikacje tworzone na podstawie jednego z oferowanych przez firmę Devart interfejsów nie wymagają instalowania dodatkowych warstw (tak jak miało to miejsce w przypadku ODBC czy BDE), dzięki czemu działają szybciej od standardowych rozwiązań.

Interfejsy te umożliwiają tworzenie aplikacji ze środowisk programistycznych Delphi, C++ Builder, Lazarus, FreePascal. Wspierają platformy VCL (Visual Components Library), FireMonkey i LCL (Lazarus Components Library). Aplikacje tworzone mogą być dla systemów Windows, Mac OS X, iOS, Android, Linux i FreBSD.

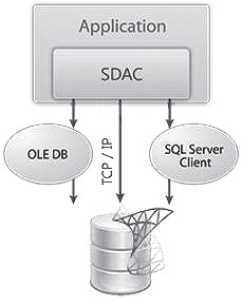
Należy wspomnieć, że firma Devart oferuje także dostawców danych dedykowanych interfejsowi ADO.NET.

Wśród zalet wyróżnić można fakt, że aplikacja nie wymaga instalacji dodatkowych warstw, ma szybsze działanie w stosunku do standardowych, uniwersalnych rozwiązań, dostępność dla wielu platform systemowych, dostęp do specjalnych funkcji serwera, możliwość użycia ponadstandardowych nowo implementowanych rozwiązań oferowanych

przez konkretne serwery bazodanowe. Wśród wad wyróżnić zaś można: niską skalowalność aplikacji w przypadku zaistnienia konieczności zmiany serwera bazodanowego i utratę elastyczności i uniwersalności aplikacji związaną ze zmianą serwera.

2.9.1. SDAC-SQL Server Data Access Components

Interfejs ten oferuje natywne połącznie z serwerem MS SQL. Ponadto oferuje dostęp do wielu funkcji, jakie realizuje serwer MS SQL Server. Zapewnia kompatybilność względem wersji SQL Server 7 [SDAC; SDAC overview].

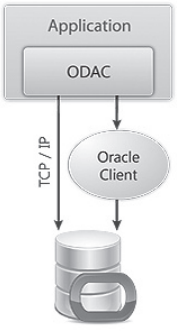


Rys. 10. Architektura SDAC

Źródło: [https://www.devart.com/sdac/].

2.9.2. ODAC − Oracle Data Access Components

Interfejs realizuje natywne połącznie z serwerem Oracle. Oferuje połączenie z serwerem Oracle z użyciem Oracle Client poprzez Oracle Call Interface lub dostęp bezpośredni za pomocą TCP/IP. Realizuje dostęp do wielu funkcji oferowanych przez serwer Oracle. Zapewnia kompatybilność do wersji Oracle 7.3 [Mościcki, Kruk 2006; ODAC; ODAC overview].

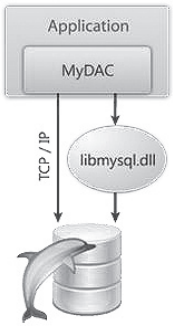


Rys. 11. Architektura ODAC

Źródło: [https://www.devart.com/odac/].

2.9.3. MyDAC − MySQL Data Access Components

Interfejs realizuje natywne połącznie z serwerem MySQL. Oferuje połączenie z serwerem MySQL z użyciem biblioteki klienta libmysql lub dostęp bezpośredni za pomocą TCP/IP. Realizuje dostęp do szerokiej gamy funkcji oferowanych przez serwer MySQL. Zapewnia kompatybilność do wersji MySQL 3.23 [MYDAC; MYDAC overview].

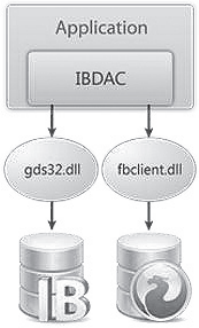


Rys. 12. Architektura MyDAC

Źródło: [https://www.devart.com/mydac/].

2.9.4. IBDAC − InterBase Data Access Components

Interfejs realizuje natywne połącznie z serwerem InterBase i Firebird. Oferuje połączenie z serwerami InterBase i Firebird z użyciem bibliotek klienckich gds32 i fb_client. Realizuje dostęp do wielu funkcji oferowanych przez serwery InterBase i Firebird. Zapewnia kompatybilność do wersji InterBase 5.x i FireBird 1.x [IBDAC; IBDAC overview].

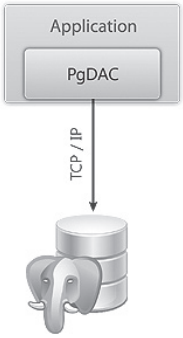


Rys. 13. Architektura IBDAC

Źródło: [https://www.devart.com/ibdac/].

2.9.5. PgDAC − PostgreSQL Data Access Components

Interfejs realizuje natywne połącznie z serwerem PostgreSQL. Oferuje bezpośrednie połączenie za pomocą TCP/IP z serwerem PostgreSQL, bez użycia biblioteki klienckiej. Realizuje dostęp do wielu funkcji oferowanych przez serwer PostgreSQL. Zapewnia kompatybilność do wersji PostgreSQL 7.1 [PGDAC; PGDAC overview].

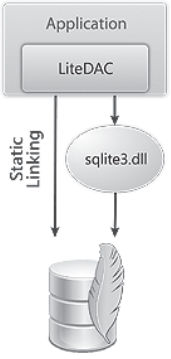


Rys. 14. Architektura PgDAC

Źródło: [https://www.devart.com/pgdac/].

2.9.6. LiteDAC − Lite Data Access Components

Interfejs realizuje natywne połącznie z serwerem SQLite. Oferuje połączenie za pośrednictwem natywnej biblioteki klienckiej lub też bezpośredniego dostępu za pomocą statycznego linkowania biblioteki klienta w aplikacji. Zapewnia dostęp do wielu funkcji oferowanych przez serwer SQLite [LITEDAC; LITEDAC overview].



Rys. 15. Architektura LiteDAC

Źródło: [https://www.devart.com/litedac/].

3. Zestawienie wybranych cech interfejsów

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

Uwzględnione zostały w nim 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	BDE	✓	–	–	–	–	–
2	BDP.NET	✓	–	–	–	–	–
3	dbGO	✓	✓	–	–	–	–
4	dbExpress	✓	✓	✓	✓	✓	✓
5	IBX	✓	✓	–	–	–	–
6	FireDAC	✓	✓	✓	–	✓	✓
7	JDBC	✓	✓	✓	✓	–	✓
8	JDO	✓	✓	✓	✓	–	–

Źródło: opracowanie własne.

Tabela 2. Wykaz lat powstania interfejsów

Lp.	Interfejs	Rok	Producent
1	BDE	1995	Borland/Inprise Corporation/Embarcadero
2	BDP.NET	2003	Borland/Codegear
3	dbGO	1999	Borland/Embarcadero
4	dbExpress	2000	Borland/Embarcadero
5	IBX	1999	Borland/Embarcadero
6	FireDAC	2007/2013	DA SOFT/Embarcadero
7	JDBC	1996	Sun/Oracle
8	JDO	2002	Oracle

Źródło: opracowanie własne.

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

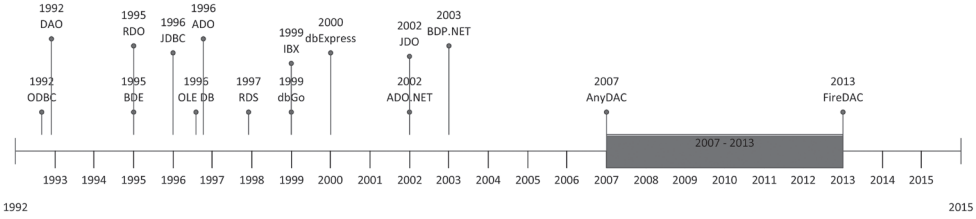
Lp.	Interfejs	Niezależność od systemu operacyjnego	Niezależność od języka programowania	Niezależność od bazy danych
1	BDE	–	✓	✓
2	BDP.NET	–	✓	✓
3	dbGO	–	✓	✓
4	dbExpress	✓	✓	✓
5	IBX	–	✓	–
6	FireDAC	✓	✓	✓
7	JDBC	✓	–	✓
8	JDO	✓	–	✓

Źródło: opracowanie własne.

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

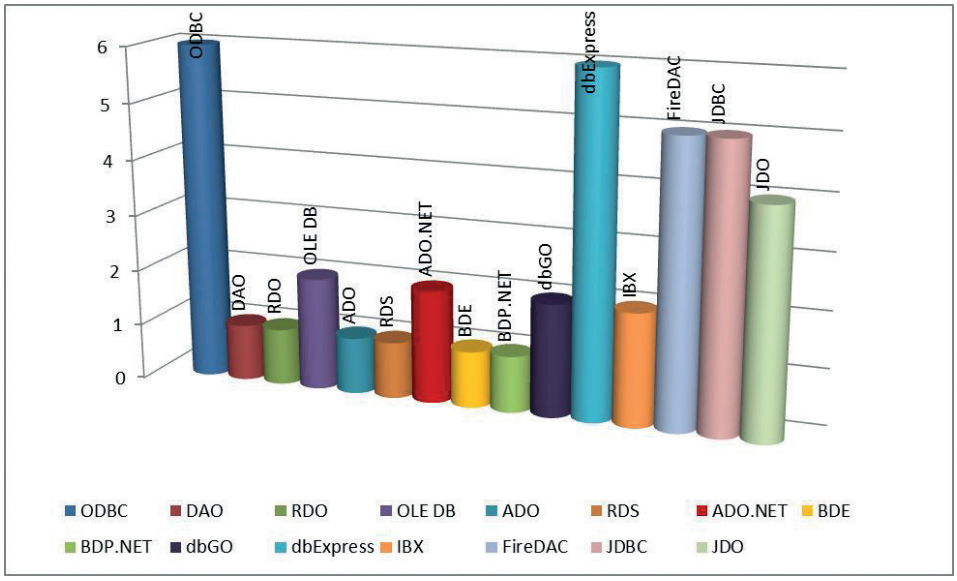
Lp.	Interfejs	Obsługa plików	Obsługa baz plikowych	Obsługa baz SQL	Obsługa baz nierelacyjnych
1	BDE	–	✓	✓	–
2	BDP.NET		✓	✓	–
3	dbGO	✓	✓	✓	–
4	dbExpress	–	–	✓	–
5	IBX	–	–	✓	–
6	FireDAC	–	✓	✓	✓
7	JDBC	✓	✓	✓	✓
8	JDO	✓	✓	✓	✓

Źródło: opracowanie własne.



Rys. 16. Wykres czasu opracowania poszczególnych interfejsów dostępu do baz danych

Źródło: opracowanie własne.



Rys. 17. Liczba platform systemowych, w obrębie których mogą współpracować wybrane interfejsy dostępu do baz danych

Źródło: opracowanie własne.

4. Zakończenie

W artykule zaprezentowano i porównano wybrane interfejsy dostępu do baz danych, opracowane przez firmę Borland, z których część dalej rozwijana była przez firmę Embarcadero, oraz rozwiązania firmy Sun Microsystems, aktualnie rozwijane pod skrzydłami Oracle. Pokaźną grupę interfejsów, podobnie jak Microsoft, opracowała firma Borland, jednakże dokonała tego w krótszym czasie. Po kilku latach dorobek ten został zwiększony przez wprowadzenie nowego interfejsu FireDAC, co miało miejsce już po wykupieniu narzędzi programistycznych Borlanda przez Embarcadero.

Przełomowym rozwiązaniem − jak na tamte czasy − było opracowanie interfejsu BDE, który łączył w sobie najlepsze cechy ODBC i IDAPI, dodatkowo wprowadzając nowe rozwiązanie w postaci bezpośrednich sterowników SQL Links, dla najpopularniejszych serwerów baz danych. Był to pierwszy uniwersalny silnik bazodanowy. Także tutaj po raz pierwszy został wprowadzony bufor rekordów. Po opracowaniu przez Microsoft technologii ADO Borland zaimplementował jego odpowiednik w swoich narzędziach pod nazwą dbGO. W tym samym roku wprowadził interfejs IBX oferujący specjalizowany dostęp do serwera InterBase. Pozwalał on uzyskać największą szybkość działania w komunikacji z serwerem InterBase oraz możliwość użycia

ponadstandardowych, nowo implementowanych funkcjonalności oferowanych przez serwer. Kolejnym godnym uwagi działaniem Borlanda było wprowadzenie zaledwie rok później nowej i szybkiej technologii dbExpress oferującej dostęp do serwerów SQL. Aktualnie największą wydajność i zarazem uniwersalność oferuje promowana przez Embarcadero technologia FireDAC. Za jej pośrednictwem nawiązywane jest połączenie z wieloma wiodącymi bazami danych oraz uzyskiwany jest dostęp do zaawansowanych właściwości oferowanych przez konkretne serwery baz danych, a to wszystko z użyciem natywnych sterowników. Nie bez znaczenia jest fakt, iż aplikacje współpracujące z bazami danych są aplikacjami w pełni natywnymi, które działają w wielu systemach operacyjnych. Aplikacje te tworzone są z jednej bazy kodu i nie wymagają żadnej dodatkowej warstwy do swojego działania (są skompilowanymi, w pełni wykonywalnymi plikami), jak ma to miejsce w przypadku aplikacji tworzonych w środowisku Java i .NET, co również dodatkowo, dodatnio wpływa na szybkość działania aplikacji.

Odmienne podejście w stosunku do rozwiązań oferowanych przez Microsoft i Borlanda zaproponowała firma Sun Microsystems, wprowadzając na rynek wzorowany na ODBC mechanizm dostępu do danych JDBC dostępny z poziomu języka Java. Oferuje on dostęp do różnych baz danych, dla których tylko dostępne są sterowniki. Stworzone aplikacje są w stanie pracować pod kontrolą wielu systemów operacyjnych, jednakże do swojego działania wymagają dodatkowej warstwy pośredniej w postaci JVM.

Na rysunku 16 zaprezentowano rozwój poszczególnych interfejsów w ostatnim ćwierćwieczu.

Spośród opisanych w artykule interfejsów BDE, BDP.NET, dbGo i IBX współpracują tylko z platformą sytemu Windows. Niekwestionowanymi liderami są JDBC oraz dbExpress i FireDAC, które są w stanie działać pod kontrolą różnych systemów operacyjnych. Liczba platform, na jakich są w stanie działać wybrane interfejsy, została zilustrowana w postaci wykresu na rys. 17.

Wszystkie zaprezentowane w tab. 3 interfejsy, poza IBX, cechuje niezależność od platformy bazodanowej.

JDBC i JDO są ściśle powiązane ze środowiskiem programowania. Praktycznie każda z omawianych technologii, poza BDP.NET, jest zależna od środowiska programistycznego. Kwestią dyskusyjną są pozostałe technologie Borlanda, które są dostępne z poziomu języka Delphi i C++ Builder, a niektóre z Lazarusa.

Wszystkie omówione technologie obsługują relacyjne bazy danych. Natomiast za pomocą interfejsu FireDAC oraz JDBC i JDO możliwy jest również dostęp do nierelacyjnych baz danych. Technologie dbExpress i IBX jako jedyne oferują tylko dostęp do serwerów SQL, co bezpośrednio wynika z ich specyfiki oraz celu, jaki przyświecał ich twórcom.

Aktualnie najbardziej popularne metody dostępu do baz danych to ADO.NET Microsoftu (dzięki platformie NET aplikacja tworzona za pomocą różnych języków uzyskuje dostęp do jednego systemu operacyjnego), JDBC − obecnie rozwijany i promowany przez Oracle (dzięki warstwie pośredniej JVM aplikacja tworzona za

pomocą jednego języka uzyskuje dostęp praktycznie do wszystkich platform systemowych, które są w stanie obsłużyć JVM). Spośród technologii oferowanych przez Embarcadero najczęściej używanymi są dbExpress wraz z mechanizmami opracowanymi (na bazie dbExpress) przez firmę Devart oraz najnowsza technologia FireDAC. Obie technologie Embarcadero mogą być użyte w aplikacjach tworzonych z poziomu dwóch języków: Delphi i C++ Builder. W przeciwieństwie do konkurencji, każda z technologii pozwala tworzyć w pełni natywne (bez żadnych warstw pośrednich) aplikacje działające pod kontrolą systemu operacyjnego Windows, Mac OS X, iOs, Android, a w przypadku dbExpress − również Linuxa.

Artykuł ten wraz z poprzednim artykułem [Łacheciński 2015], który w całości poświęcony został interfejsom autorstwa Microsoft, stanowi cykl, będący podstawą do dalszych badań w odniesieniu do wydajności, a także możliwości oferowanych przez poszczególne interfejsy. Wyniki badań uzyskane na podstawie serii testów praktycznych zostaną przedstawione w kolejnym artykule, stanowiącym trzecią część cyklu.

Literatura

BDE, http://edn.embarcadero.com/article/28688 (24.10.2015).

BDE, dbGo, DBX, http://theprofessionalspoint.blogspot.com/2013/09/what-is-difference-between--dbexpress.html (25.10.2015).

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, pl/kkntpd/tpd_pliki/publikacja/.

BDP.NET, http://edn.embarcadero.com/article/31939 (24.10.2015).

Cantu M., 2003, *Delphi 7. Praktyka programowania,* tom II, Mikom, Warszawa.

Connolly T., Begg C., 2004, *Systemy baz danych. Praktyczne metody projektowania, implementacji i zarządzania,* tom 2, RM, Warszawa.

Darakhvelidze P., Markov E., 2005, *Delphi. Techniki bazodanowe i internetowe*, Helion, Gliwice.

DBX, http://docwiki.embarcadero.com/RADStudio/Seattle/en/Deploying_dbExpress_Database_Applications (25.10.2015).

FIBPlus, http://www.devrace.com/en/fibplus/ (19.11.2015).

FireDAC, http://www.embarcadero.com.pl/produkty/firedac/ (26.10.2015).

FireDAC DocWiki, http://docwiki.embarcadero.com/RADStudio/Seattle/en/Overview_(FireDAC).

http://edn.embarcadero.com/article/31939.

http://www.developersbook.com/jdbc/interview-questions/jdbc-interview-questions-faqs.php.

http://www.embarcadero.com.pl/produkty/firedac/.

https://docs.oracle.com/cd/B14099_19/web.1012/b15901/sessions012.htm.

https://www.devart.com/ibdac/.

https://www.devart.com/litedac/.

https://www.devart.com/mydac/.

https://www.devart.com/odac/.

https://www.devart.com/pgdac/.

https://www.devart.com/sdac/.

IBDAC, https://www.devart.com/ibdac/ (29.10.2015).

IBDAC overview, https://www.devart.com/ibdac/docs/ (29.10.2015).

IBO, http://delphibag.com/sol/ibobjects.html (19.11.2015).
IBO Codegear, http://cc.codegear.com/partners/interbase7/cps_computer_programming_solutions/ib_objects/index.html (19.11.2015).
IBO Objects, http://www.ibobjects.com/ibowhatis.html (19.11.2015).
IBX: http://docwiki.embarcadero.com/RADStudio/XE8/en/Getting_Started_with_InterBase_Express (26.10.2015).
JDBC DevelopersBook, http://www.developersbook.com/jdbc/interview-questions/jdbc-interview-questions-faqs.php.
JDBC docs, http://docs.oracle.com/javase/tutorial/jdbc/overview/index.html (5.11.2015).
JDBC Stanford, http://infolab.stanford.edu/~ullman/fcdb/oracle/or-jdbc.html (5.11.2015).
JDBC technetwork overview, http://www.oracle.com/technetwork/java/overview-141217.html (5.11.2015).
JDBC technetwork, http://www.oracle.com/technetwork/java/javase/jdbc/index.html (5.11.2015).
JDO, https://db.apache.org/jdo/why_jdo.html (12.11.2015).
JDO docs, https://docs.oracle.com/cd/B14099_19/web.1012/b15901/sessions012.htm (12.11.2015).
JDO Service architecture, http://www.service-architecture.com/articles/database/java_data_objects_jdo.html (12.11.2015).
JDO technetwork: http://www.oracle.com/technetwork/java/index-jsp-135919.html (12.11.2015).
LITEDAC, https://www.devart.com/litedac/ (29.10.2015).
LITEDAC overview, https://www.devart.com/litedac/docs/?overview.htm (29.10.2015).
Łacheciński S., 2015, *Interfejsy dostępu do baz danych – przegląd technologii Microsoft*, 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.
Mościcki A., Kruk I., 2006, *Oracle 10g i Delphi. Programowanie baz danych*, Helion, Gliwice.
MYDAC, https://www.devart.com/mydac/ (29.10.2015).
MYDAC overview, https://www.devart.com/mydac/docs/?overview.htm (29.10.2015).
ODAC, https://www.devart.com/odac/ (29.10.2015).
ODAC overview, https://www.devart.com/odac/docs/?overview.htm (29.10.2015).
PGDAC, https://www.devart.com/pgdac/ (29.10.2015).
PGDAC overview, https://www.devart.com/pgdac/docs/?overview.htm (29.10.2015).
SDAC, https://www.devart.com/sdac/ (29.10.2015).
SDAC overview, https://www.devart.com/sdac/docs/?overview.htm (29.10.2015).
SQLJ, JDBC, http://docs.oracle.com/cd/B28359_01/java.111/b31227.pdf (5.11.2015).
Tyagi S., McCammon K., Vorburger M., Bobzin H., 2004, *Java Data Objects*, Helion, Gliwice.
Wojciechowski M., Zakrzewicz M., 2002, *Analiza porównawcza technologii tworzenia aplikacji internetowych dla baz danych Oracle*, VIII konferencja PLOUG, Kościelisko.
Wybrańczyk M., 2004, *Delphi 7 i bazy danych*, Helion, Gliwice.