Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

PRZEGLĄD METOD BUDOWY APLIKACJI UŻYTKOWYCH DLA BAZ DANYCH ORACLE

Artur Gramacki

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: a.gramacki@iie.uz.zgora.pl

STRESZCZENIE

W artykule omówiono najpopularniejsze metody tworzenia aplikacji użytkowych dla systemu zarządzania bazą danych (SZBD) Oracle. Oddzielnie omówiono metody tworzenia aplikacji internetowych (obecnie jest to bardzo dynamicznie rozwijająca się klasa aplikacji) oraz tzw. aplikacji samodzielnych (w architekturze dwuwarstwowej typu klient-serwer).

1. WPROWADZENIE

Obecnie bez trudu można zauważyć ogromny postęp w dziedzinie poprawy jakości działania sieci Internet. Fakt ten został oczywiście szybko zauważony przez twórców systemów zarządzania bazami danych (SZBD), w tym oczywiście przez firmę Oracle.

Można bez wielkiego ryzyka stwierdzić, że Internet jest i będzie w przyszłości podstawowym medium transmisyjnym dla wszelkiego rodzaju danych przechowywanych w systemach klasy SZBD. Konieczne jest więc posiadanie wiedzy na temat dostępnych technologii wytwarzania aplikacji internetowych. Omówieniu tychże poświęcony jest niniejszy artykuł.

Istnieją oczywiście całkiem spore obszary (np. sektor bankowy, ubezpieczeniowy), gdzie samodzielne aplikacje klienckie mają ciągle ogromne zastosowanie i nie jest konieczne oraz ekonomicznie i merytorycznie uzasadnione migrowanie do środowiska internetowego. Z drugiej jednak strony widać coraz większą chęć otwarcia się na Internet, czego efektem jest np. stale powiększająca się oferta banków internetowych (np. mBank). Pod pojęciem „samodzielne aplikacje klienckie" rozumieć będziemy aplikacje nie wymagające dostępu do Intrenetu, choć zwykle korzystają one z jakiejś formy lokalnej sieci komputerowej.

Metody wytwarzania tego typu aplikacji są bardziej ustabilizowane i nie następują tutaj tak wielkie zmiany, jak to ma miejsce w przypadku aplikacji internetowych. Te również zostaną pokrótce omówione w dalszej części artykułu.

2. METODY WYTWARZANIA APLIKACJI KLIENCKICH

2.1. Prekompilator Pro*C/C++

[1] Jak powszechnie wiadomo języki C oraz C++ są obecnie jednymi z podstawowych narzędzi do tworzenia wszelkiego rodzaju oprogramowania. W tych to językach powstają bardzo zaawansowane współczesne aplikacje. Z kolei w dziedzinie baz danych niekwestionowanym liderem jest język SQL. Nie jest to jednak typowy język programowania – należy raczej mówić, iż jest to rodzaj interpretera do formułowania zapytań do relacyjnej bazy danych. W SQL-u nie jest możliwe zakodowanie jakiegokolwiek algorytmu, gdyż brak jest w nim konstrukcji sterujących typowych dla „zwykłych" języków programowania. Nie jest to absolutnie wada SQL-a, gdyż został on stworzony do zupełnie innych zadań i tam sprawdza się znakomicie.

Aby więc można było z poziomu języków C/C++ wykorzystać potężne możliwości języka SQL, konieczne jest stworzenie mechanizmów umożliwiających umieszczanie w kodach źródłowych C/C++ wywołań SQL-owych. To zadanie możliwe jest właśnie dzięki prekompilatorowi Pro*C/C++. Jego istota działania polega na tym, że plik źródłowy C/C++ z umieszczonymi w nim „wstawkami" SQL-owymi zostaje przetworzony (wstępnie prekompilowany) w taki sposób, że wstawki te zostają zamienione na wywołania odpowiednich funkcji bibliotecznych i inne konstrukcje zgodne z językami C/C++. Następnie tak przetworzony kod jest kompilowany w zwykły sposób.

Zalety: Szybkość działania i stabilność.

Wady: W zasadzie brak (pod warunkiem bardzo dobrego opanowania programowania w językach C/C++). Konieczność poznania dużej liczby funkcji API, tzw. funkcji OCI (ang. *Oracle Call Interface*).

2.2. Interfejs ODBC

Interfejs ODBC (ang. *Open Database Connectivity*) to znany już od wielu lat standard zapewniający dostęp (w ujednolicony sposób) do dowolnej bazy danych. Posiadając odpowiedni sterownik ODBC (zwykle dostarczany przez twórcę bazy danych) możemy praktycznie z poziomu każdego języka programowania połączyć się z bazą danych i pobrać lub zmodyfikować interesujące nas dane. Co więcej wiele gotowych programów ma zaimplementowaną obsługę standardu ODBC. Przykładowo z poziomu programu Microsoft Excel można kilkoma kliknięciami myszki wyświetlić dane z interesujących nas tabel bazy Oracle. Praktycznie wszystkie obecnie liczące się na rynku zintegrowane środowiska programistyczne zapewniają wsparcie dla ODBC. Można tu wymienić choćby takie narzędzia jak Borland Delphi, Microsoft Visual C++, Microsoft Visual Basic.

Zalety: Ogólnie przyjęty standard.

Wady: Brak wsparcia dla bardzo popularnego języka JAVA. Technologia JDBC poza podobną nazwą nie ma wiele wspólnego z ODBC.

2.3. Pakiet Developer 6i oraz 9i

[1] Pakiet Developer to narzędzie autorskie firmy Oracle skierowane do tych użytkowników, którzy znają język PL/SQL (jest to wbudowany w bazę Oracle, jej wewnętrzny język programowania). Jest to zintegrowane środowisko programistyczne umożliwiające tworzenie aplikacji klienckich. Do ich uruchamiania wymagane jest wcześniejsze zainstalowanie na komputerach użytkowników tzw. *runtime-u*, czyli niewielkiego objętościowo oprogramowania, umożliwiającego uruchomienie utworzonej w pakiecie Developer aplikacji. Ponieważ aplikacje kodowane są w języku PL/SQL, który jest bardzo silnie osadzony w systemie Oracle, dlatego też tworzone w pakiecie Developer programy są w zasadzie nieprzenaszalne do innych systemów bazodanowych.

Najnowsza wersja 9i wprowadza bardzo istotną zmianę: odchodzi się mianowicie od samodzielnych aplikacji typu *runtime*. W zamian za to można je uruchamiać w przeglądarkach internetowych (jako bardzo rozbudowane, automatycznie generowane, aplety) i w związku z tym pakiet w tej wersji można uznać za narzędzie do tworzenia aplikacji internetowych.

Zalety: Doskonała integracja z bazą ORACLE.

Wady: Rozwiązanie tylko dla bazy Oracle, konieczność programowania w nieprzenoszalnym do innych systemów bazodanowych języku PL/SQL, nieco ubogie możliwości projektowania interfejsu użytkownika (np. brak bardzo popularnych obiektów typu g*rid*).

2.4. Język Java

[2] Język JAVA w połączeniu z protokołami JDBC (standard firmy Sun) oraz SQLJ (standard ANSI/ISO) umożliwia tworzenie w pełni funkcjonalnych aplikacji bazodanowych typu klienckiego. Bogactwo bibliotek, funkcji systemowych oraz obiektowość mogą być bardzo zachęcające dla programistów, którzy preferują techniki programowania obiektowego, zamiast bardziej klasycznego programowania strukturalnego (moda, czy konieczność?). Obecnie praktycznie każdy producent baz danych dostarcza do swoich systemów odpowiednie sterowniki JDBC i/lub SQLJ. Oba standardy mają swoje wady i zalety, oba oferują przenaszalność aplikacji oraz podobną funkcjonalność. Standard SQLJ umożliwia sprawdzanie poprawności instrukcji SQL już na etapie kompilacji oraz daje prostszy i bardziej zwarty kod źródłowy. Ponadto zapewnia binarną przenaszalność aplikacji między różnymi systemami baz danych. JDBC jest natomiast bardziej elastyczny niż SQLJ. Wydaje się, że obecnie bardziej popularny jest JDBC, choć oczywiście w przyszłości może się to zmienić.

Zalety: Bogactwo bibliotek, wielka popularność języka.

Wady: Duże aplikacje robią wrażenie „ociężałych" w porównaniu do stworzonych w innych językach programowania. Jest to jednak wada (miejmy nadzieję, że przemijająca) wszystkich programów pisanych w języku JAVA.

3. METODY WYTWARZANIA APLIKACJI INTERNETOWYCH

3.1. Skrypty CGI

[4] CGI (ang. *Common Gateway Interface*) jest specyfikacją interfejsu pomiędzy serwerem WWW a programami użytkownika. Programy te nazywa się właśnie skryptami CGI. W odróżnieniu od statycznych stron HTML przechowywanych po stronie serwera WWW, skrypty CGI są w stanie wygenerować w trybie *on-line* stronę HTML i za pośrednictwem serwera WWW odesłać ją do przeglądarki użytkownika. Innymi słowy CGI to rodzaju mechanizm umożliwiający uruchamianie po stronie serwera WWW dowolnego programu, przy czym polecenie jego uruchomienia dociera do serwera za pośrednictwem przeglądarki internetowej użytkownika. Skrypty CGI można pisać praktycznie w każdym języku programowania, jak np. C/C++, Perl, ASP, JSP, Visual Basic, Pascal, skrypty powłoki UNIX itd. Wybór konkretnego języka jest w zasadzie kwestią upodobań programisty.

Zalety: Prostota. Ogromna liczba wspieranych języków programowania. Działa na praktycznie każdym serwerze WWW.

Wady: Nie zapewnia wystarczającej skalowalności. Przy dużej liczbie odwołań do skryptu CGI może dojść do zawieszenia całego serwera WWW.

3.2. Serwlety Javy oraz Java Server Pages (JSP)

[2] Serwlety oraz JSP to aplikacje napisane w języku Java, które uruchamiane są po stronie serwera WWW a ich zadaniem jest generowanie dynamicznych stron HTML odsyłanych następnie do przeglądarki internetowej. Dostęp do baz danych, podobnie jak w przypadku innych aplikacji w języku Java odbywa się w oparciu o standardy JDBC oraz SQLJ.

Różnica pomiędzy serwletami a JSP jest istotna w zasadzie tylko dla programisty, gdyż z punktu widzenia serwera WWW jest to jedno i to samo. Mianowicie przy pierwszym odwołani do pliku JSP jest on automatycznie tłumaczony do postaci równoważnego serwletu i każde kolejne odwołanie do dokumentu JSP powoduje uruchomienie tego właśnie serwletu.

Z punktu widzenia programisty istnieją pewne różnice w stylu programowania. Nie wnikając w tej chwili w szczegóły można powiedzieć tylko tyle, że kod JSP jest bardziej zwarty niż odpowiadający mu kod w postaci serwletu. Tworząc serwlet „zanurzamy" w kodzie Javy kod HTML, natomiast w przypadku dokumentów JSP jest dokładnie odwrotnie: wewnątrz statycznych stron HTML umieszczamy odpowiednie „wstawki" języka JAVA (używając tzw. biblioteki znaczników – ang. *tag library* – można tworzyć dokumenty JSP nie zawierające żadnych fragmentów kodu w języku Java).

Zalety: Oparte o język Java. W przypadku dokumentów JSP stosunkowo dobra separacja kodu Javy od statycznego kodu HTML. Część technologii J2EE, która jest tworzona pod kątem możliwości budowania bardzo złożonych aplikacji biznesowych. Zdecydowanie lepsza skalowalność niż skrypty CGI.

Wady: Rozwlekłość kodów źródłowych. W przypadku serwletów trudne czasami do opanowania przemieszanie kodów odpowiedzialnych za tworzenie części statycznych i dynamicznych stron HTML.

3.3. Procedury składowane PL/SQL oraz PL/SQL Server Pages (PSP)

[1] Tworzenie aplikacji internetowych w języku PL/SQL polega na tym, że za pomocą odpowiednich pakietów (też napisanych w PL/SQL) możliwe jest wywoływanie z poziomu przeglądarki internetowej procedur i funkcji PL/SQL zeskładowanych w bazie danych. Za pomocą odpowiednich procedur (podstawowe znaczenie mają tutaj procedury *htp.print* i *htp.prn*) możliwe jest wysyłanie do przeglądarki poleceń HTML przekazanych im jako argument. W ten sposób można w łatwy sposób generować dynamiczne strony HTML. Ponadto wewnątrz kodu PL/SQL można osadzać kod SQL co zapewnia nam pełną kontrolę nad zgromadzonymi danymi.

Ponieważ aplikacje internetowe pisane w języku PL/SQL w żaden sposób nie oddzielają kodów HTML od logiki aplikacji tworzonej w języku PL/SQL i SQL, przeto firma Oracle zaproponowała rozwiązanie nieco podobne do JSP a mianowicie możliwość osadzania w plikach HTML (za pomocą odpowiednich znaczników) fragmentów kodów w PL/SQL. W efekcie programista tworzy dokumenty PSP, które następnie muszą być załadowane do bazy danych za pomocą narzędzia o nazwie *loadpsp*. Narzędzie to w swej istocie tłumaczy dokument PSP na równoważny mu kod PL/SQL i składuje go w bazie danych na zwykłych zasadach.

Zalety: Technologia łatwa do opanowania dla znających języki SQL oraz PL/SQL. Duża szybkość działania. Doskonała integracja z bazą Oracle.
Wady: Technologia ograniczona tylko do baz danych Oracle. Pewne trudności w budowaniu aplikacji o bardzo złożonej logice i przewadze stron HTML o charakterze statycznych nad dynamicznymi.

3.4. Język skryptowy PHP

[3] Język PHP „od zawsze" służył do tworzenia aplikacji internetowych. Stosunkowo późno został on dostrzeżony przez firmę Oracle, która do wersji 8i nie zapewniała dla niego wsparcia (co gorsza próba włączania na własną rękę modułu *mod_php* do serwera Apache, na którym bazuje Oracle HTTP Server, powodowała utratę wsparcia technicznego!). Wraz jednak z promowaniem przez firmę środowiska linuksowego, musiało wcześniej lub później dojść do otwarcia na PHP, gdyż język ten jest w tych środowiskach zdecydowanie na pierwszym miejscu popularności (nawet w porównaniu do Javy). Istota programowania w PHP polega na umiejętnym osadzaniu w kodzie HTML wstawek PHP, których zadaniem jest łączenie się z bazą Oracle, pobieranie z niej interesujących nas danych i przedstawianie ich na stronach WWW. PHP wspiera dwa różne interfejsy do łączenia z bazą Oracle: standardowy (*oracle*) oraz bardziej elastyczny i dający więcej możliwości (*oci8*). Pewną niedogodnością języka PHP jest niewątpliwie to, że pracując z różnymi bazami danych korzystamy z osobnego zestawu funkcji. Na pewno nie ułatwia to przenaszalności aplikacji pomiędzy różnymi systemami bazodanowymi.

Zalety: Prostota i wielka popularność języka. Bogactwo bibliotek. Technologia darmowa.
Wady: Różny zestaw funkcji dla różnych baz danych.

3.5. Pakiet HTML DB

[1] Omawiany tu pakiet jest bardzo nowym (i dlatego w ocenie autora mało dojrzałym) rozwiązaniem firmy Oracle. Jest to uruchamiany z poziomu przeglądarki internetowej interakcyjny system umożliwiający w sposób „myszkologiczny" tworzenie aplikacji internetowych. Niewątpliwą zaletą tego rozwiązania jest to, że nawet osoba mająca bardzo niewielkie doświadczenie w programowaniu aplikacji dla bazy Oracle w „normalnych" językach programowania, może w miarę szybko stworzyć dość ładnie wyglądającą aplikację o podstawowej funkcjonalności. Niestety, to co jest zaletą dla niewprawnego programisty, jest dużą wadą dla kogoś bardziej doświadczonego. Taka osoba z pewnością niezbyt chętnie sięgnie po to narzędzie. Ilości okienek, które trzeba przejść, aby zbudować nawet najprostszą aplikację wystawia na ciężką próbę cierpliwość programisty. Niemniej jednak wspominamy tutaj o tym systemie, gdyż może on w przyszłości (po dopracowaniu) być dość wygodnym narzędziem do szybkiego prototypowania i/lub tworzenia interfejsu aplikacji interenetowych. Wydaje się jednak, że cała logika tej aplikacji powinna być tworzona w klasycznych językach programowania – w przypadku bazy Oracle będzie to na pewno język PL/SQL i/lub Java.

Zalety: Względna prostota, umożliwiająca w miarę szybko otrzymanie działającej aplikacji internetowej o podstawowej funkcjonalności.

Wady: Uciążliwa (choć dość prosta) obsługa. Wielka trudność w dopasowaniu aplikacji (wygląd, funkcjonalność) do specyficznych potrzeb klienta. Brak przenaszalności do innych systemów bazodanowych.

4. ZAKOŃCZENIE

W artykule omówiono najważniejsze zdaniem autora techniki tworzenia aplikacji użytkowych korzystających z bazy danych Oracle. Wiele z omówionych technik może być stosowanych również w innych niż Oracle systemach. Te dedykowane wyłącznie dla bazy Oracle mają tą zaletę, że tworzone z ich pomocą oprogramowanie cechuje duża stabilność i szybkość działania. Oczywiście wadą jest tutaj nieprzenaszalność do innych SZBD. Wybór odpowiedniej technologii jest w dużej mierze kwestią upodobań i znajomości danego narzędzia przez programistę. Istotny jest również charakter aplikacji – czy ma w dużej części charakter statyczny, czy też bardzo intensywnie korzysta z bazy danych. Nie bez znaczenia są również środki finansowe, którymi dysponujemy. Reasumując trudno jest tutaj wskazać zdecydowanego faworyta. Ostateczny wybór zawsze musi uwzględniać charakter i wielkość tworzonej aplikacji.

LITERATURA

[1] http://otn.oracle.com
[2] http://java.sun.com
[3] http://www.php.net
[4] http://www.apache.org