BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 17, 2002

Internetowe aplikacje bazodanowe

Grzegorz KROPIEWNICKI

Zakład Teleinformatyki, Instytut Automatyki i Robotyki WAT
ul. Kaliskiego 2, 00-908 Warszawa

STRESZCZENIE: W artykule przedstawiono zagadnienia związane z wytwarzaniem internetowych aplikacji bazodanowych z wykorzystaniem języka skryptowego PHP oraz serwera bazy danych PostgreSQL.

1. Wstęp

Sieć Internet staje się standardowym medium wykorzystywanym do przekazywania i udostępniania informacji. Od pewnego czasu daje się zauważyć postępującą integrację różnych usług w ramach wspólnego środowiska WWW. Posługując się przeglądarką można dziś sprawdzić pocztę elektroniczną, czy przeglądać pliki na serwerach ftp. Integracja ta dotyczy również udostępniania przez WWW informacji gromadzonych w bazach danych, które staje się ważnym aspektem Internetu. Wiele witryn internetowych wykorzystuje mechanizmy dostępu do baz danych, w których gromadzone są informacje, pozwalając na ich wygodne przeglądanie, wyszukiwanie, aktualizowanie oraz poszerzanie o nowe informacje. Sprzężenie stron WWW z bazą danych stwarza możliwość tworzenia rozbudowanych serwisów informacji dając z jednej strony to, co oferuje sam system bazy danych - zarządzanie i organizację informacji oraz z drugiej strony to, co daje środowisko WWW - atrakcyjny, wygodny i dobrze znany użytkownikom interfejs.

2. Projektowanie bazy danych

Właściwe zaprojektowanie relacyjnej bazy danych ma ogromne znaczenie dla całego systemu udostępniania informacji. Przeniesienie rzeczy lub zdarzeń, które ma modelować baza, w relacje i powiązania między tymi rzeczami lub zdarzeniami stanowi pierwszy etap tworzenia całej aplikacji udostępniającej gromadzone dane użytkownikom. Dobrze przemyślana i poprawna struktura relacji przyczynia się do zachowania integralności danych przy równoczesnym zachowaniu możliwości przyszłej rozbudowy bazy danych. Prawidłowa struktura bazy może też w znacznym stopniu ułatwić zaprogramowanie do niej interfejsu użytkownika.

Istnieje kilka sposobów tworzenia projektu bazy danych. Pamiętać jednak należy, że celem najwyższym jest stworzenie projektu bazy danych, której pielęgnacja będzie jak najłatwiejsza i którą inne osoby będą mogły modyfikować.

Poprawnie zbudowana baza danych pozwala wykonać następujące czynności:

- dodawać nowe kolumny do istniejących tabel;
- tworzyć nowe tabele;
- zrozumieć przeznaczenie wszystkich tabel oraz ich kolumn;
- szybko tworzyć aplikacje użytkownika;
- wprowadzać zmiany w bazie danych bez konieczności jednoczesnego wprowadzania poważnych zmian w kodzie stron WWW;
- szybko odczytywać informacje przechowywane w tabelach.

W celu zapewnienia wysokiej jakości projektu bazy danych używa się kilku powszechnie znanych technik. Należą do nich między innymi normalizacja bazy danych i stosowanie tabel słownikowych.

2.1. Normalizacja bazy danych

Normalizacja ma na celu uporządkowanie tablic, tak aby były one maksymalnie wydajne oraz zwarte. Najczęściej popełnianym błędem jest gromadzenie niepotrzebnych danych. Technika normalizacji pozwala określić i usunąć z bazy danych nadmiarowe dane. Podczas fazy normalizacji sprawdza się zgodność struktur bazy danych z postaciami normalnymi oraz poddaje się je poprawkom, jeżeli zgodność ta nie jest zachowana. Postać normalna to zestaw kryteriów, które musi spełniać dana tabela, aby mogła być uznana za poprawną i nie przyczyniała się do powstawania błędów. Istnieje kilka stopni normalizacji. Im wyższy stopień tym mniej niepotrzebnych informacji będzie zawierała baza. Jednocześnie skala trudności projektowania bazy danych rośnie wraz z jej

stopniem. Poszczególne stopnie normalizacji określają sposób przechowywania danych:

I. Pierwsza postać normalna. Warunkiem osiągnięcia pierwszej postaci normalnej jest to, aby każde pole tabeli zawierało unikalne dane. Jeśli w tabeli bazy znalazłyby się dwie kolumny zawierające tą samą daną np. nazwisko osoby, to taka baza nie osiągnęłaby nigdy pierwszej postaci normalnej. Proste usunięcie nadmiarowej kolumny mogłoby sprawić, że baza danych spełni wymagania pierwszej postaci normalnej.

II. Druga postać normalna. W tej postaci bazy żadne pole w tabeli nie może zawierać danych, które można wyznaczyć na podstawie pozostałych pól tabeli. Jeśli kolumna tabeli bazy zawiera daną, którą można określić na podstawie danych zapisanych w innych kolumnach, to baza danych nie spełnia wymogów drugiej postaci normalnej.

III. Trzecia postać normalna. W tym przypadku dane nie mogą powtarzać się w obrębie całej bazy danych. Można podać tu przykład bazy danych, w której przechowywane będą dane o klientach i ich zamówieniach. W pierwszej kolejności tworzy się tabelę klientów zawierającą:
    - identyfikator klienta, który jest wartością unikalną,
    - imię i nazwisko klienta,
    - sposób rozliczenia,
    - adres wysyłki.

    Następnie tworzy się tabelę zamówień, zawierającą:
    - nazwę zamawianego towaru ,
    - identyfikatora klienta, który umożliwia skojarzenie rekordów tej tabeli z rekordem tabeli klientów,
    - imię i nazwisko klienta,
    - sposób rozliczenia,
    - adres wysyłki.

    W kolejnym kroku tworzymy powiązanie pomiędzy tymi tabelami z wykorzystaniem kolumn zawierających identyfikator klienta. W takim przypadku kolumny: nazwisko klienta, sposób rozliczania oraz adres wysyłki w tabeli zamówień są danymi nadmiarowymi, co jest niezgodne z trzecią postacią normalną. Dlatego też operacja usunięcia nadmiarowych kolumn pozwoli spełnić bazie danych wymogi trzeciej postaci normalnej.

Należy zawsze przeprowadzać normalizację bazy danych. Zalecane jest przeprowadzanie normalizacji do trzeciej postaci normalnej, a konieczne jest doprowadzenie bazy do drugiej postaci normalnej. Istnieją także wyższe stopnie normalizacji, jednak nie są one często stosowane.

2.2. Użycie tabel słownikowych

Tworząc projekt bazy danych pamiętać należy o głównym założeniu: konieczne jest stworzenie bazy cechującej się łatwością pielęgnacji i modyfikacji. W celu uzyskania prostoty bazy danych należy zadbać, aby jej struktura była przejrzysta i zrozumiała.

W celu zapewnienia jednoznacznych rozwinięć identyfikatorów lub skrótów używanych w tabelach bazy danych najlepszą metodą jest użycie tabeli słownikowej. Dodatkowo tabele słownikowe wykorzystywane są do zapewnienia integralności bazy. Ograniczają one zakres wartości używanych w innych tabelach bazy danych do konkretnego zbioru wartości. Tabele słownikowe zazwyczaj składają się z dwóch pól: jedno pole zawiera identyfikator lub skrót, drugie stanowi rozwinięcie opisując wartości pierwszego pola. Przykładem tabeli słownikowej jest tabela zawierająca opis tytułów naukowych i zawodowych:

Tytuł	Opis tytułu
inż.	Inżynier
mgr	Magister
dr	Doktor
hab.	Doktor habilitowany
prof.	Profesor

Rys. 1. Przykład tabeli słownikowej

Dane w tabelach słownikowych rzadko ulegają zmianie i równie rzadko wprowadza się do nich nowe rekordy lub usuwa istniejące.

3. Realizacja internetowej aplikacji bazodanowej

Realizację aplikacji internetowej należy rozpocząć od wyboru odpowiedniego serwera bazy danych. Dopiero potem można przystąpić do opracowania struktur w bazie danych do przechowywania danych i stworzenia interfejsu użytkownika. Przy projektowaniu bazodanowej aplikacji internetowej należy uwzględnić fakt, iż w protokole HTTP brak jest mechanizmów kontroli stanu. Dlatego też należy zadbać o mechanizm przechowywania i przekazywania identyfikatora użytkownika (sesji) między poszczególnymi stronami aplikacji. Ważną cechą aplikacji internetowej jest zapewnienie autoryzacji dostępu i poufności przesyłanych danych.

3.1. Wybór serwera bazy danych

Na rynku istnieje duży wybór serwerów bazy danych. Wykorzystanie produktów komercyjnych daje między innymi gwarancję wsparcia technicznego ze strony producenta, jednak wiąże się ze znacznymi kosztami licencji. Rozwiązania oparte na przykład o Microsoft SQL Server pracujący na platformie Windows2000, czy produkty takich firm jak Oracle, Informix, IBM czy Sybase można często wykorzystać tylko przy zastosowaniach komercyjnych. Do wyboru jest również wiele profesjonalnych systemów komercyjnych pracujących pod kontrolą systemu operacyjnego Linux, np. Empress. Wielu innych producentów baz danych ma w planach lub już udostępniło wersję swoich systemów na SO Linux. Istnieje również cała gama serwerów baz danych udostępnionych, podobnie jak sam system Linux, na zasadach *Open Source.* Jednym z najsilniejszych mocno zaawansowanych, darmowych, relacyjnych systemów baz danych dla SO Linux jest PostgreSQL.

3.2. Charakterystyka PostgreSQL

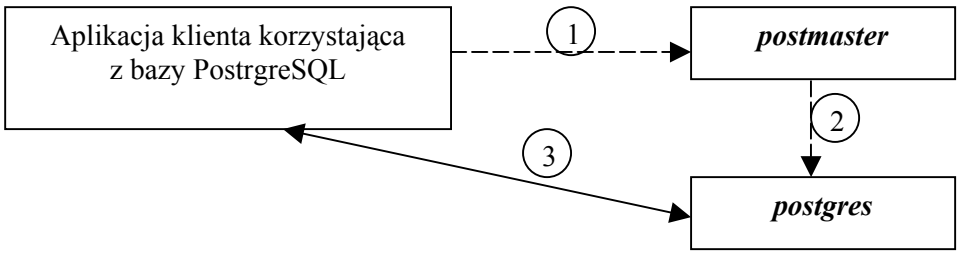
PostgreSQL wywodzi się z projektu relacyjnego systemu bazy danych Postgres zapoczątkowanego w 1986 roku na Uniwersytecie Kalifornijskim Berkeley przez prace prof. Michaela Stonebrakera. Pierwsza wersja systemu zaprezentowana została w 1987 roku.

PostgreSQL jest systemem zarządzania bazami danych zaliczanym do systemów obiektowo-relacyjnych. Dysponuje rozbudowaną listą obsługiwanych typów oraz szeregiem funkcji wbudowanych i zagregowanych. Realizuje niemal wszystkie zaawansowane mechanizmy uchodzące dziś za standardy w dużych, komercyjnych systemach baz danych. PostgreSQL wdraża między innymi procedury wyzwalane (wyzwalacze, ang. *triggers)* stanowiące kod użytkownika przechowywany wewnątrz bazy i uruchamiany w określonych sytuacjach: przed wykonaniem zapytania aktualizującego dane, usuwającego i wprowadzającego nowe dane do bazy. Działając na najniższym poziomie dostępu do bazy, wyzwalacze są bardzo ważnym narzędziem pozwalającym utrzymać integralność referencyjną bazy. Innym mechanizmem jest ochrona integralności danych oraz transakcje, w których użytkownik może pracować na kopii bazy danych do momentu zatwierdzenia transakcji lub anulować wszystkie modyfikacje. PostgreSQL, podobnie jak wiele innych SZRBD, pozwala na tworzenie perspektyw (ang. *views*). Stanowią one wcześniej zdefiniowane i przechowywane wewnątrz bazy danych instrukcje wybierające określone dane. Pozwalają więc na zapamiętanie i późniejsze wygodne wywoływanie często zadawanych, złożonych zapytań. Dodatkowo istnieje możliwość dynamicznego ładowania prekompilowanych napisanych zwykle w języku C bibliotek definiujących funkcje lub nowe typy danych oraz tworzenia wstawek w kilku

oferowanych językach proceduralnych. W celu umożliwienia przechowywania w bazie binarnych danych o dużych rozmiarach, takich jak pliki czy dane multimedialne, PostgreSQL posiada bardzo wygodny mechanizm dostępu do tych danych *(Large Object Interface),* działający w podobny sposób jak funkcje dostępu do plików w systemie UNIX.

3.3. Architektura systemu PostgreSQL

Serwer danych PostgreSQL opiera się na architekturze typu klient/serwer, w której każdemu użytkownikowi przydzielany jest odrębny proces serwera. Z tego względu konieczny jest jeden, wspólny proces nadzorujący *(postmaster),* który na każde żądanie klienta wywołuje proces serwera *(postgres).* Proces *postmaster* jest uruchamiany w systemie jako demon i jest związany z określonym w konfiguracji portem TCP/IP, na którym oczekuje wywołań. Jest też odpowiedzialny za usuwanie procesów oraz zarządzanie przydzielaniem im pamięci.



Rys. 2. Kroki komunikacji z serwerem bazy danych PostrgreSQL.
1 – nawiązanie połączenia aplikacji klienta z procesem postmaster,
2 – powołanie przez postmaster procesu postgres do obsługi żądań klienta,
3 – ustanowienie połączenia do dalszej komunikacji.

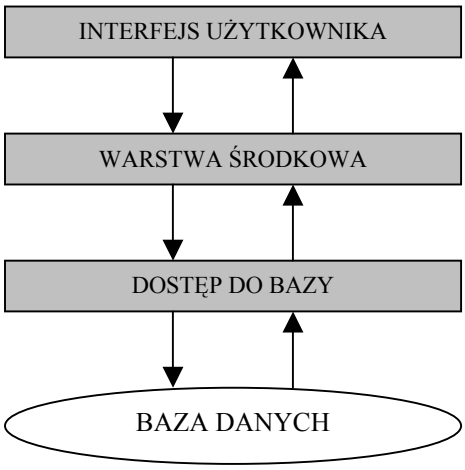
Dla większości systemów baz danych instrukcje SQL wywoływane przez proces klienta mogą być przekazywane do programu sterującego na trzy sposoby:

1. Przekazywanie zapytań z klawiatury lub pliku do interpretatora SQL. W PostgreSQL jest to program *psql.* Stanowi on główne narzędzie administracji bazami danych PostgreSQL. Zapytania przesyłane są w formacie tekstowym ASCII, a rezultat zwracany przez serwer jest przesyłany tym samym, ustanowionym wcześniej połączeniem.

2. Osadzanie instrukcji SQL w innym języku (np. C). Tak napisany program jest przed kompilacją we właściwym mu języku poddawany prekompilacji przez program interpretujący osadzone instrukcje SQL i zamieniający je na wywołania funkcji języka macierzystego.
3. Wykorzystanie interfejsu poziomu wywołań CLI (ang. *SQL Call-Level lnterface).* Program realizujący dostęp do systemu implementuje wówczas serię funkcji odpowiadających rozkazom SQL. Funkcje te znajdują się wewnątrz bibliotek dołączanych do pakietu baz danych. Dla systemu PostgreSQL biblioteką taką jest *libpq* (w języku C) oraz *libpg++* (w języku C++).

3.4. Metody dostępu przez WWW

Internetowa aplikacja korzystająca z bazy danych ma za zadanie odizolować użytkownika od samej bazy danych i umożliwić wykonywanie ściśle określonych operacji przetwarzających przechowywane informacje w sposób możliwie prosty i funkcjonalny.



Rys. 3. Model trójwarstwowy dla aplikacji bazodanowej

Architektura internetowej aplikacji bazodanowej oparta jest o model trójwarstwowy (rys. 3), który wprowadza logiczny, funkcjonalny podział aplikacji na trzy warstwy. Najniższa warstwa odpowiedzialna jest za dostęp do bazy danych, odbieranie instrukcji SQL z warstwy środkowej i zwracanie

wyników lub informacji o wystąpieniu błędów. Zawarte są w niej sterowniki i oprogramowanie sieciowe umożliwiające zdalną komunikację z serwerem bazy danych. W warstwę środkową wbudowuje się reguły dziedziny danych zgodnie, z którymi przetwarzane są instrukcje przekazywane z warstwy górnej. Warstwa najwyższa, interfejs użytkownika, odpowiada za komunikację z użytkownikiem. Jej zadaniem jest prezentacja danych oraz komunikatów o błędach.

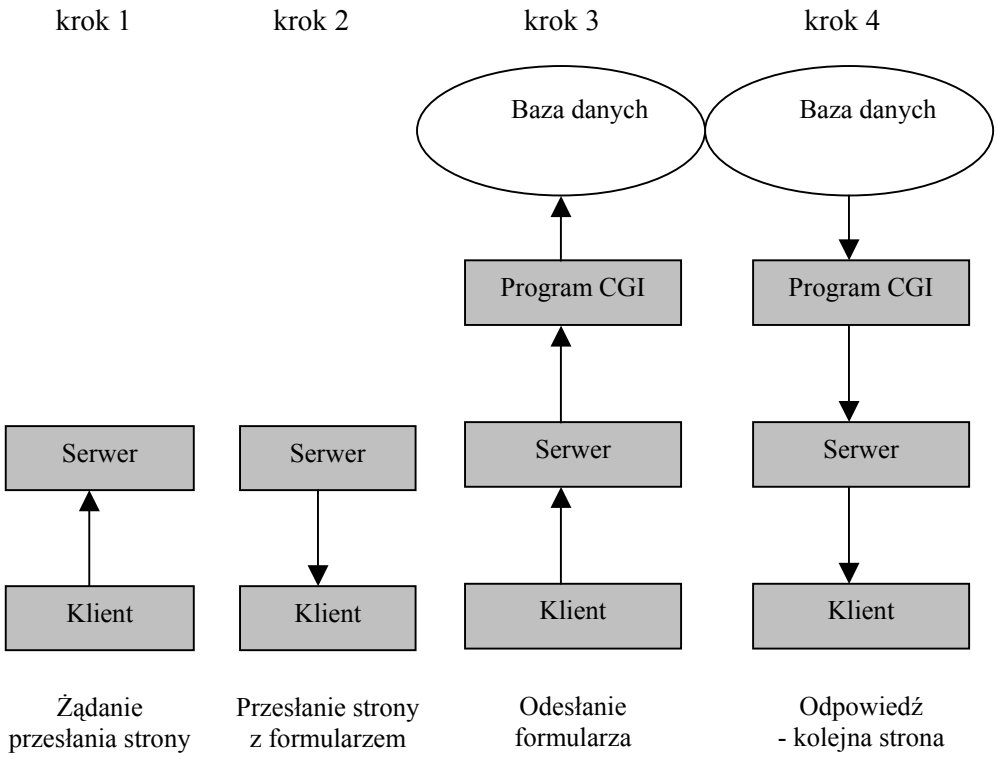
W aplikacjach sieciowych dąży się zwykle do przeniesienia możliwie największego ciężaru aplikacji na stronę serwera, pozostawiając klientowi zazwyczaj jedynie funkcje interfejsu z użytkownikiem. W takiej sytuacji aplikacja jest łatwiejsza w utrzymaniu – większość lub całość kodu ulegającego modyfikacji związanej ze zmianami w bazie, rozbudową lub poprawkami skupiona jest tylko w części serwerowej aplikacji.

Ogromną zaletą zastosowania przeglądarki internetowej jako środowiska aplikacji bazy danych spełniającego zadanie warstwy interfejsu użytkownika jest możliwość skorzystania z gotowych, graficznych możliwości formatu HTML. Format ten oferuje dość bogaty zestaw technik i elementów umożliwiających tworzenie atrakcyjnego i funkcjonalnego interfejsu. Ponadto w architekturze takiej kod aplikacji umieszczony jest w jednej kopii na serwerze WWW. Jeśli jest to ten sam komputer, który stanowi serwer bazy danych, wówczas wszelkie modyfikacje aplikacji dokonywane są w tym samym miejscu. Oprogramowanie klienta tzn. przeglądarka internetowa pozostaje niezmienione. Jedynym ograniczeniem nakładanym na aplikację są więc możliwości przeglądarki klienta, które jednak mogą być w znacznym stopniu rozszerzane, na przykład przez stosowanie apletów Javy lub JavaScriptu. Sam program przeglądarki internetowej stanowi więc dla aplikacji środowisko, w którym zrealizowana jest górna warstwa modelu tzn. warstwa interfejsu.

Zdalny dostęp do bazy danych przez sieć WWW może być zrealizowany na wiele sposobów. Klasyczne rozwiązanie opiera się na wykorzystaniu dołączonych do pakietu serwera WWW bibliotek funkcji CLI do stworzenia programów, z którymi poprzez interfejs CGI komunikuje się przeglądarka internetowa klienta.

Stroną inicjującą połączenie jest klient przez wpisanie adresu URL. W następnym kroku odbywa się przesłanie użytkownikowi strony zawierającej żądanie przesłania formularza (krok 2), do którego użytkownik wpisuje dane. Przesłany formularz jest odbierany przez określony w formularzu program CGI, który z kolei komunikuje się z bazą danych (krok 3). W odpowiedzi program CGI generuje kod HTML ładowany do przeglądarki klienta jak każda inna, statyczna strona WWW (krok 4). Wygenerowana strona może oczywiście zawierać nowe formularze odwołujące się do kolejnych programów CGI. Poważną wadą tego schematu komunikacji aplikacji klienta z serwerem bazy

danych jest brak interakcji z użytkownikiem. Program CGI jest uruchamiany tylko po przesłaniu formularza i po zwróceniu wyniku w postaci kodu HTML jest zamykany. Program CGI zawsze musi więc czekać na przesłanie formularza i nie ma pamięci poprzednich wywołań.



Rys. 4. Schemat komunikacji klienta z serwerem wykorzystującym mechanizm CGI

Innym rozwiązaniem przetwarzania żądań klienta są aplety Javy. Możliwości Javy dające się wykorzystać przy programowaniu apletów pozwalają na tworzenie nawet bardzo złożonych aplikacji wykorzystujących dużo bardziej zaawansowane techniki niż oferowane przez format HTML. Oferują one całą gamę obiektów graficznych czy technikę przeciągnij-upuść. Aplety Javy wykorzystują interfejs JDBC do komunikacji z serwerem bazy danych. Mają też możliwość użycia gniazdka do otwarcia połączenia z serwerem, z którego zostały załadowane. Jednak złożone aplety mają często pokaźne rozmiary, co znacznie spowalnia ich ładowanie do przeglądarki klienta. Poza tym przy aplikacjach, z których korzysta równocześnie wielu

użytkowników, możliwości apletów związane z tworzeniem stałych połączeń z serwerem okazują się wadą ze względu na wydajność aplikacji.

Jednym z najwygodniejszych i najszybszych sposobów tworzenia aktywnych stron WWW jest stosowanie skryptów wykonywanych po stronie serwera (np.: PHP). Idea polega na przeplataniu kodu HTML strony z instrukcjami języka skryptowego, które zawiera się w odpowiednich znacznikach, wyróżniających je od kodu HTML. Przed wysłaniem strony do klienta jej kod jest przetwarzany przez serwer, który interpretuje osadzone polecenia skryptu. Wynikiem działania interpretera jest czysty kod HTML przesyłany do przeglądarki klienta. Rozwiązanie takie ma kilka niebagatelnych zalet. Do najważniejszych można zaliczyć:

1. Bezpieczeństwo danych i procedur. Poprzez umieszczenie aplikacji i danych na serwerze można w znacznie lepszy sposób kontrolować dostęp do nich. Żaden klient nie będzie w stanie dostać się do programów służących do odczytywania i manipulowania danymi.
2. Bezpieczeństwo komunikacji. Dzięki możliwości wykorzystania SSL i protokołu HTTP, język skryptowy może zagwarantować bardzo wysokie bezpieczeństwo danych i zabezpieczyć je przed „podsłuchiwaniem".
3. Łatwość zarządzania. Aplikacje działające po stronie serwera są zazwyczaj łatwiejsze do zarządzania i aktualizacji. W takim przypadku istnieje tylko jeden komputer, na którym znajduje się program. Do klienta przesyłany jest wyłącznie kod HTML interpretowany przez wszystkie przeglądarki internetowe bez względu na złożoność zwracającego go skryptu. Kod aplikacji jest więc całkowicie niewidoczny dla klienta, a dynamicznie tworzone strony są nie do odróżnienia od stron statycznie zapisanych na serwerze WWW.
4. Efektywność działania. Aplikacje w dużym stopniu wykorzystujące bazy danych działają tym lepiej, im „bliżej" serwera bazy danych są uruchamiane.

Jeśli język taki wyposaży się w możliwości komunikacji z serwerem danych, narzędzie takie doskonale nadaje się do tworzenia interfejsu WWW dla baz danych. Do komunikacji klient/serwer wykorzystuje się te same techniki, z których korzystają programy CGI - formularze oraz interfejs CGI.

Na rynku istnieje wiele narzędzi działających w opisany sposób. Do najbardziej znanych należy język ASP *(ang. Active Server Pages)* firmy Microsoft oraz PHP (*ang. PHP Hypertext Preprocesor*)**.** PHP jest narzędziem darmowym i doskonale współpracującym z serwerem Apache na Linuxie. ASP jest produktem komercyjnym dobrze zintegrowany z systemem Windows, umożliwiając programiście korzystanie z potężnej technologii jaką stanowi

COM+. Typowe aplikacje baz danych wykorzystujące ASP realizują warstwę środkową aplikacji w postaci wielu oddzielnych komponentów COM, które z jednej strony mają dostęp do środowiska skryptu ASP i bezpośrednio mogą zwracać kod HTML, a z drugiej strony mogą komunikować się z bazą danych. Możliwości języka PHP w niczym nie ustępują rozwiązaniom komercyjnym.

Skrypty PHP są przetwarzane w sposób wsadowy. Wpływa to korzystnie na wydajność serwera, ale wymusza stosowanie dodatkowych mechanizmów pozwalających na zapamiętanie stanu aplikacji i przenoszenie tych informacji pomiędzy kolejnymi wywołaniami stron PHP oraz identyfikowanie żądań pochodzących od tego samego klienta. Tworzenie i zarządzanie sesjami użytkowników jest problemem wszystkich złożonych aplikacji działających w środowisku WWW i wynika z bezstanowej natury protokołu HTTP, gdzie kolejne wywołania nie są ze sobą w żaden sposób powiązane i każde jest traktowane przez serwer osobno. Problem zapamiętania stanu w aplikacji internetowej zostanie omówiony w dalszej części opracowania.

3.5. Charakterystyka PHP

PHP został zapoczątkowany w 1994 roku przez Rasumsa Lerdorfa. Pierwsza wersja była używana przez autora w jego prywatnych stronach WWW do śledzenia odwiedzin użytkowników. W 1995 roku grono użytkowników tego programu powiększyło się. PHP był wówczas znany jako *Personal Home Page Tools* i stosował kilka prostych instrukcji oraz wyrażeń. W roku 1995 Rasums Lerdorf przepisał kod programu twor

(ang. *FormInterpreter)* pochodzi
danych z formularzy HTML. W
SZBD mSQL. Udostępnienie P
bardzo szybkim rozwojem aplik
szacowano na 15000, w 1997 jęz
kod PHP został ponownie pr
Gutmansa twor

automatyczna inicjalizacja pewnych zmiennych w każdym programie PHP. Automatycznie zainicjowane i gotowe do użycia w kodzie są wszystkie zmienne środowiskowe serwera oraz wszystkie dane z formularzy przekazane zarówno metodą POST jak i GET. Możliwości tworzenia klas i styl związany z cechami obiektowymi przypomina z kolei język Java. Wbudowane funkcje PHP pozwalają tworzyć nawet najbardziej złożone aplikacje, a ich zbiór ciągle się poszerza dzięki otwartości źródeł PHP udostępnionych w sieci. Stosunkowo łatwo jest też dodawać własne funkcje napisane w języku C. Możliwości PHP są ogromne - od manipulacji nagłówkami HTTP, obsługi protokołów IMAP, SMNP, POP3 i NNTP, poprzez programowe tworzenie grafiki w formacie GIF i dokumentów PDF, do obsługi ciasteczek i zarządzania sesjami użytkowników. Jedną z najmocniejszych stron PHP jest jednak obsługa wielu serwerów baz danych, w tym również pakietów komercyjnych. Lista obsługiwanych SZBD jest spora: Dbase, Empress, Informix, MySQL, Oracle, PostgreSQL, MS SQL Server, Sybase i wiele innych. PHP jest potężnym i funkcjonalnym narzędziem. Przy jego pomocy można uzyskać pełny dostęp do systemu plików, wykonywać polecenia systemowe i otwierać połączenia sieciowe. Cechy te z jednej strony stwarzają olbrzymie możliwości dla twórców aktywnych serwisów WWW, z drugiej stanowią potencjalne źródło niebezpieczeństwa przy niewłaściwej konfiguracji. PHP.

3.6. Problem identyfikacji sesji

Twórcom protokołu HTTP zależało na tym, żeby był on tak szybki, jak to tylko możliwe i dlatego nie zaimplementowano w nim żadnych mechanizmów kontroli stanu. Z tego powodu HTTP nazywamy protokołem **bezkontekstowym** lub **bez kontroli stanu**. Między odrębnie wywołanymi sesjami protokołu nie zachodzą żadne relacje. Serwer WWW nie dysponuje mechanizmem rozróżniania pojedynczych użytkowników i nie posiada żadnych informacji o ich sesjach. Aby móc przetwarzać takie informacje, trzeba opracować własną metodę ich pozyskiwania. Pod pojęciem sesji rozumiemy każde odwiedzenie witryny WWW związane z przeglądaniem jednej lub kilku stron. Przykładowa sesja zakupów w sklepie internetowym może składać się z: dodania towaru do koszyka, przejścia na stronę kontrolną, wprowadzenia adresu i parametrów karty kredytowej, złożenia zamówienia i zamknięcia okna przeglądarki.

Do kojarzenia danych z użytkownikiem potrzebny jest numer identyfikacyjny sesji. W pierwszym podejściu wydawać się może, że naturalnym identyfikatorem mógłby być adres IP komputera użytkownika. Istnieje wiele wad takiego rozwiązania, które ostatecznie każe z niego zrezygnować:

- liczni dostawcy usług internetowych wymuszają na klientach łączących się telefonicznie korzystanie z serwera pośredniczącego. Do serwera

WWW dostarczony jest adres IP pośrednika. Jeżeli w tym samym czasie z aplikacji zechce skorzystać dwóch użytkowników tego samego pośrednika może nastąpić konflikt adresów i brak jednoznacznego odwzorowania.
- niektórzy dostawcy usług internetowych zmieniają co jakiś czas adresy IP swoich klientów, by uniemożliwić im np. podłączenia do sieci własnych serwerów WWW.
- w sieciach z dynamicznym przydziałem adresów IP po zamknięciu połączenia w trakcie kontynuowania np.: zakupów w sklepie inetrnetowym nie mamy gwarancji otrzymania tego samego adresu IP.

Identyfikatory muszą być całkowicie losowe, bo w przeciwnym razie użytkownicy mogą próbować je odgadywać i przejmować cudze sesje, co w konsekwencji mogłoby prowadzić do przechwycenia danych np.: numeru karty kredytowej.

Istnieje kilka sposobów identyfikowania sesji użytkownika. Do najważniejszych zalicza się:
- autoryzację opartą o podstawowy schemat,
- wykorzystanie pliku cookie,
- przekazanie identyfikatora sesji w adresie URL,
- stosowanie ukrytych pól formularzy.

Jednym z najwcześniejszych mechanizmów do przechowywania stanu klienta była autoryzacja oparta o tzw. podstawowy schemat (*basic authentication*). Przy jej zastosowaniu każdy użytkownik otrzymuje swój własny identyfikator i hasło. Jej działanie polega na tym, iż serwer oczekuje, że do zlecenia będzie dołączony nagłówek *Authorization*. Jeśli go nie ma lub identyfikator i hasło są niepoprawne, wtedy zlecenie jest odrzucone. W większości przypadków, kiedy przeglądarka otrzyma odpowiedź o odrzuceniu, wyświetla na ekranie okno dialogowe pozwalające na ponowne wprowadzenie danych identyfikacyjnych, które później będą przesłane wraz z każdym następnym zleceniem dotyczącym tego serwisu.

Inna metoda powszechnie stosowana polega na przekazaniu identyfikatora za pomocą pliku cookie. Przekazywanie identyfikatora sesji przez ciasteczka (*ang. cookies*) to dla programisty najprostsze rozwiązanie. Jedyna funkcja której obsługę trzeba dodać do aplikacji, służy założeniu odpowiedniego pliku. Cookies przesyłane są jako jeszcze jeden nagłówek zlecenia lub odpowiedzi. Po zalogowaniu użytkownika i poprawnej weryfikacji serwer przesyła do przeglądarki klienta ciasteczko zawierające unikalne dane (identyfikator sesji). Cookie zostaje zapamiętane przez przeglądarkę. Przy kolejnym żądaniu użytkownika odwiedzającego serwis dane z cookies są przesyłane do serwera w nagłówku żądania. Jeśli ciasteczko zawiera poprawne dane, klient jest traktowany jako zweryfikowany. Wraz z danymi

identyfikującymi sesję, w ciasteczkach można również przesyłać inne informacje związane z użytkownikiem i stanem aplikacji takie jak identyfikator i hasło użytkownika, jego preferencje dotyczące wyglądu serwisu itp.

Poniżej przedstawiono przykładowe cookie:

Set-cookie: Session_ID=00001; path=/; domain=.mcp.com;
expires=Staurday, 01-12-2002 10:00:00 GMT

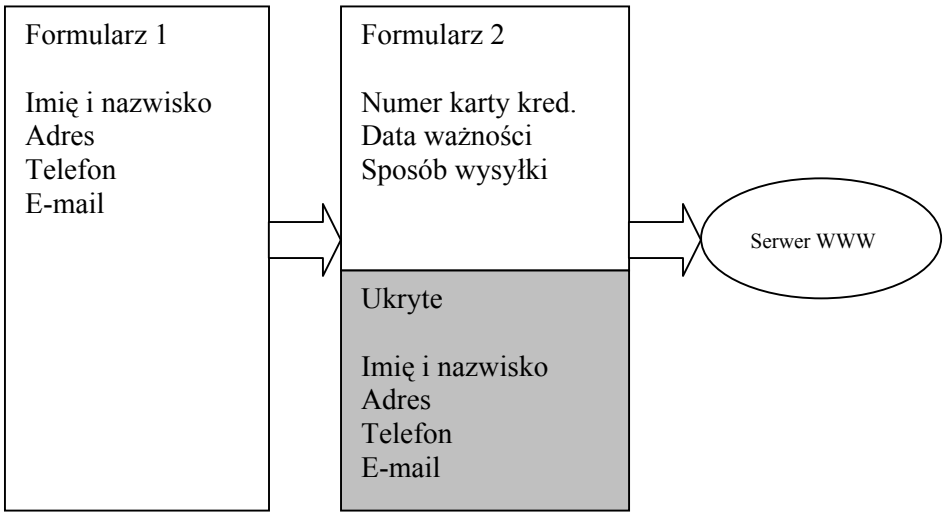
Można w nim wyróżnić pary *nazwa=wartosc*, określające parametry cookie. Zarówno nazwa jak i wartość ustalane są przez serwer. Maksymalny rozmiar pary *nazwa=wartosc* nie może przekroczyć 4kB. W przykładzie cookie ma nazwę Session_ID i wartość 00001. Pole *path* określa ścieżkę na serwerze, której dotyczy cookie. Jeśli chcemy na przykład, aby cookie dotyczyło tylko skryptów CGI, możemy ustawić ścieżkę na katalog przechowujący skrypty. Domena *domain* pozwala na określenie serwera, którego cookie dotyczy. W powyższym przykładzie przeglądarka będzie odsyłać cookie przy każdym łączeniu się z dowolnym serwerem z domeny *mcp.com*. Wykorzystując parametr *domain* można określić pojedynczy serwer. Pole *expires* określa, kiedy przeglądarka powinna automatycznie wykasować cookie. Jeśli nie zostanie określona data ważności, wówczas cookie zostanie usunięte po zamknięciu przeglądarki.

Poważnym ograniczeniem takiego scenariusza zarządzania sesją jest możliwość całkowitego wyłączenia obsługi ciasteczek przez użytkownika przeglądarki. Poza tym część starszych przeglądarek nie akceptuje ich. Ciasteczka mogą być z powodzeniem stosowane do sporządzania statystyk odwiedzin stron WWW, czy sporządzania profilu użytkownika, ale nie powinny być jednak podstawą działania aplikacji.

Z tych względów wiele aplikacji stosuje mechanizm przekazywania identyfikatora sesji w adresie URL. Taka metoda jest dostępna w wersji 4. języka PHP. Zarządzanie sesją może być wówczas sprzężone z bazą danych, w której z każdym identyfikatorem mogą być związane dowolne dane. Przykładowy scenariusz zarządzania sesjami użytkowników opierałby się na przydzieleniu zweryfikowanemu użytkownikowi pseudolosowej sekwencji znaków. W specjalnej tabeli bazy danych zapisywane byłby: przydzielony identyfikator sesji, adres IP użytkownika, jego stały identyfikator w bazie danych oraz dokładna data i czas zapisu. Przydzielony identyfikator sesji byłby następnie przekazywany w adresie URL do kolejnych stron aplikacji. Przy żądaniu kolejnej strony aplikacja najpierw odbierałaby identyfikator sesji i sprawdzała go w tabeli sesji. Sprawdzany byłby również adres IP klienta oraz data i czas otwarcia sesji. Jeśli czas otwartej sesji przekraczałby ustalony czas

trwania sesji lub dane identyfikacyjne byłby niepoprawne, użytkownik zostałby o tym poinformowany i proszony o ponowne zalogowanie. Dodatkowa weryfikacja adresu IP klienta wyklucza możliwość użycia ważnego identyfikatora sesji przez innego użytkownika.

Kolejnym sposobem identyfikacji użytkownika aplikacji jest stosowanie ukrytych pól formularzy. Polega on na wprowadzeniu do pierwszego formularza danych identyfikacyjnych, które przesyłane są do kolejnych formularzy jako pola ukryte. Rysunek 5 przedstawia w jaki sposób są przechowywane informacje w formularzach.



Rys. 5. Przetwarzanie zamówień przy pomocy dwóch formularzy i pól ukrytych

Jak widać na rysunku użycie kolejnych formularzy umożliwia podzielenie danych na logiczne części, dzięki czemu użytkownik wypełnia tylko część jawną, pozostałe pola wypełniane są przez aplikację.

4. Podsumowanie

Aplikacje bazodanowe działające w środowisku WWW są coraz powszechniejszym sposobem udostępniania informacji szerokiemu gronu użytkowników. Są łatwe i szybkie w tworzeniu oraz rozbudowie. Przy projektowaniu tego typu aplikacji należy również zwrócić uwagę na bezpieczeństwo dostępu do aplikacji jak również przesyłania danych. Do autoryzacji dostępu można wykorzystać istniejące właściwości samego serwera

WWW lub języka PHP. Do poufności danych można stosować protokół SSL (Secure Socket Layer) – protokół bezpiecznej komunikacji między klientem, a serwerem stworzony przez Netscape.

Konfiguracja oparta na systemie PostgreSQL jako serwera baz danych i dostępie realizowanym przez skrypty PHP doskonale nadaje się do tworzenia akademickich, edukacyjnych serwerów gromadzących i udostępniających dane w atrakcyjnym, dobrze znanym użytkownikom formacie sieciowym WWW.

Literatura:

[1] Rafe Colburn: *CGI.* Helion. Gliwice 1998.

[2] Tobias Ratschiller, Till Gerken: *PHP4 aplikacje.* Wydawnictwo Robomatic. Wrocław 2001.

[3] Craig Hilton, Jeff Willis: *PHP – Internetowe aplikacje bazodanowe.* Helion. Gliwice 2000.

[4] Michael J. Hernandez: *Bazy danych dla zwykłych śmiertelników.* Edu-Mikom. Warszawa 1998.

[5] www.php.com

Recenzent: dr inż. Janusz Furtak

Praca wpłynęła do redakcji 30.10.2002