

1 Wprowadzenie

Na obserwowany w ostatnich latach wzrost zainteresowania aplikacjami analitycznymi dla magazynów danych (OLAP) nakłada obecnie się znaczny wzrost popularności języka programowania Java oraz technologii wielowarstwowych. W wyniku tego, język Java coraz częściej staje się podstawową platformą implementacji aplikacji wspomagających podejmowanie decyzji. Najpopularniejsze rozwiązania aplikacyjne bazują na architekturach samodzielnych aplikacji Java, appletów Java oraz serwletów Java.

Ważnym elementem wszelkich aplikacji bazodanowych jest realizacja komunikacji z systemem zarządzania bazą danych. Programista, który dotychczas konstruował aplikacje Java dla systemów OLTP, najczęściej wykorzystywał w tym celu bibliotekę JDBC (Java Database Connectivity). JDBC stanowiła zbiór klas Java, umożliwiających nawiązanie połączenia z bazą danych, przesłanie polecenia SQL oraz odebranie wyniku jego pracy. Na bazie biblioteki JDBC powstało wiele innych rozwiązań, jak na przykład biblioteki komponentów InfoSwing, umożliwiające łatwą budowę wizualnych aplikacji dostępu do baz danych.

Filozofia JDBC może być wykorzystywana również w aplikacjach typu OLAP. Jednak ze względu na bardzo złożony charakter zapytań kierowanych do systemu magazynu danych oraz ze względu na wymagania wysokiej interakcyjności aplikacji, rośnie zapotrzebowanie na specjalizowane interfejsy programistyczne dla dostępu do magazynu danych. Odpowiedzią rynku na takie zapotrzebowanie są prace nad uniwersalnymi interfejsami nazywanymi OLAP API, dzięki którym programista mógłby w jednolity i nieskomplikowany sposób korzystać z systemów magazynów danych różnych producentów. Jedną ze zgłaszanych w tym zakresie propozycji jest Java OLAP API dla systemu zarządzania bazą danych Oracle9i. Java OLAP API jest interfejsem programistycznym umożliwiającym przetwarzanie danych w magazynie danych z poziomu programu Java. Java OLAP API może być elastycznie wykorzystywany w różnych architekturach aplikacji Java.

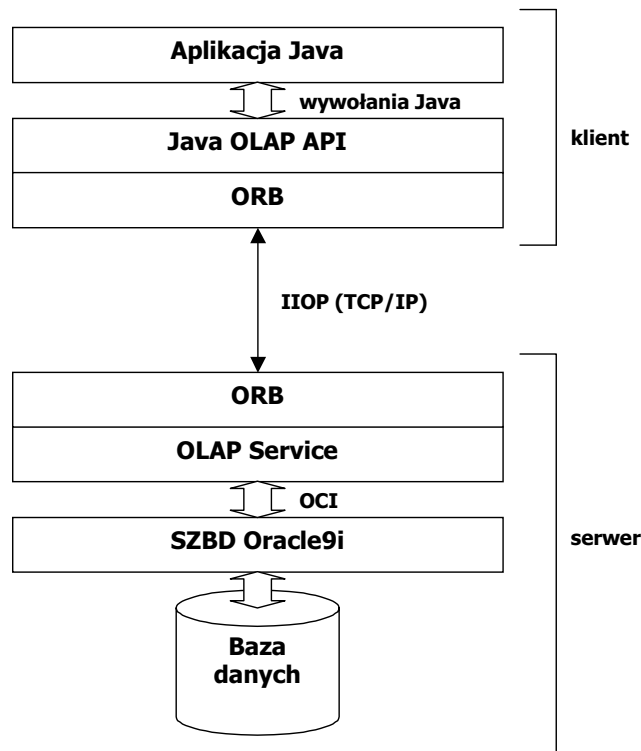
Na bazie interfejsu Java OLAP API powstała biblioteka komponentów Business Intelligence Beans, która umożliwia programiście konstrukcję wizualnych i interakcyjnych aplikacji graficznych opartych na magazynach danych zarządzanych przez Oracle9i. Dzięki budowie aplikacji z gotowych „klocków”, jakimi są komponenty Business Intelligence Beans, możemy w stosunkowo krótkim czasie konstruować funkcjonalne i elastyczne aplikacje dla wspomagania podejmowania decyzji. W pewnym sensie, komponenty Business Intelligence Beans pełnią w stosunku do Java OLAP API podobną funkcję, jak InfoSwing w stosunku do JDBC.

W artykule omówiono architekturę i funkcjonalność komponentów z biblioteki Business Intelligence Beans. W rozdziale drugim krótko opisano usługę Oracle OLAP Services. Rozdział trzeci omawia interfejs programistyczny Java OLAP API. Rozdział czwarty stanowi przegląd własności komponentów Business Intelligence Beans.

2 Usługa OLAP Services

Jedną z najbardziej zauważalnych nowości w systemie zarządzania bazą danych Oracle9i jest integracja dotychczasowego serwera relacyjnej bazy danych z serwerem wielowymiarowej bazy danych Express. Elementem integrującym obie technologie jest nowa usługa systemowa nazywana OLAP Services. OLAP Services obsługuje żądania zdalnych aplikacji, dotyczące operacji przetwarzania danych, a następnie kieruje je, odpowiednio, do bazy relacyjnej lub wielowymiarowej. Z punktu widzenia aplikacji użytkownika, usługa OLAP Services obsługuje otrzymane żądania w jednolity sposób, niezależnie od fizycznej reprezentacji danych (relacyjnej ROLAP lub wielowymiarowej MOLAP).

3 Java OLAP API

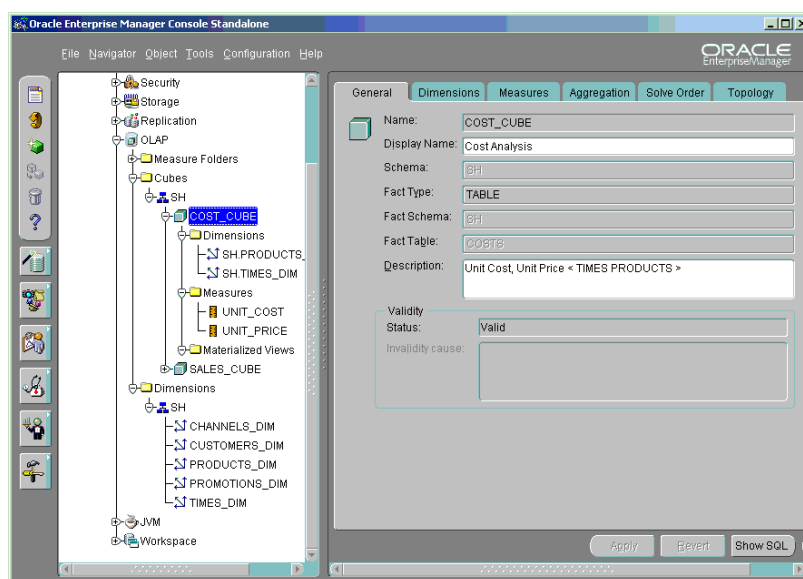


Rys.1 Ogólna architektura aplikacji analitycznej opartej na Java OLAP API

Przebieg komunikacji aplikacji użytkownika z bazą danych jest następujący. W pierwszej kolejności, aplikacja wywołuje odpowiednią metodę interfejsu Java OLAP API, przekazując jej żądanie pobrania danych z bazy danych. Następnie, interfejs Java OLAP API generuje stosowne żądanie przesłania sieciowego i przekazuje je do modułu ORB (Object Request Broker ze specyfikacji CORBA). W kolejnym kroku, moduł ORB przesyła treść żądania w ramce protokołu IIOP (Internet Inter-ORB Protocol) do usługi Oracle OLAP Service. Wtedy oprogramowanie OLAP Service generuje treść zapytania SQL odpowiadającego żądaniu pobrania danych z bazy danych. Na zakończenie, zapytanie SQL jest przekazywane do wykonania instancji Oracle9i, a wynik zapytania wraca taką samą drogą do aplikacji użytkownika.

Implementacja Java OLAP API wymaga wcześniejszego przygotowania następujących struktur magazynu danych w bazie danych Oracle9i:

1. Tabeli faktów i tabel wymiarów, zorganizowanych w strukturę gwiazdy lub płotka śniegu; struktury te mogą zostać przygotowane przy użyciu podstawowych poleceń SQL.
2. Metadanych opisujących model logiczny struktur istniejących w bazie danych: miar, wymiarów i kostek wielowymiarowych; metadane powinny zostać zdefiniowane przy użyciu narzędzia Oracle Enterprise Manager (rys. 2).



Rys. 2. Definicja metadanych w narzędziu Oracle Enterprise Manager

Z punktu widzenia programisty, implementacja interfejsu Java OLAP API jest dostarczana w formie dziesięciu plików bibliotecznych JAR: `collections.jar`, `express_common.jar`, `express_connection.jar`, `express_mdm.jar`, `express_olapi_common.jar`, `express_olapi_data.jar`, `express_olapi_data_full.jar`, `express_olapi_data_receiveOnly.jar`, `express_olapi_indep.jar`, `express_spl.jar`. Oprócz tego, do prawidłowej pracy Java OLAP API wymagana jest obecność klas `VisiBrokera` (`vbjorb.jar`, `vbjapp.jar`, `vbjtools.jar`) oraz klas CORBA wytwarzanych przez Oracle (`aurora_client.jar`, `aurora_client_orbdep.jar`, `aurora_client_orbindep.jar`).

W celu zilustrowania specyfiki interfejsu Java OLAP API, poniżej przedstawiono treść kodu źródłowego, który spowoduje pobranie z magazynu danych informacji na temat tych produktów, które w miesiącu maju 2002 zarobiły w Wielkopolsce więcej niż 100000 zł. Dla porównania przedstawiono także treść analogicznego kodu w języku Express.

Java OLAP API:

```
Source regionSel = geography.selectValue("Wielkopolska");
Source czasSel = time.SelectValue("MAY2002");
Source zlotychZaProdukt = zlotychSrc.join(regionSel).join(czasSel);
Source prodSel = product.select(zlotychZaProdukt.gt(100000));
Source wynik = zlotychSrc.join(regionSel).join(czasSel).join(prodSel);
```

Express:

```
limit geography to 'Wielkopolska'
limit time to 'MAY2002'
limit product to zlotych gt 100000
```

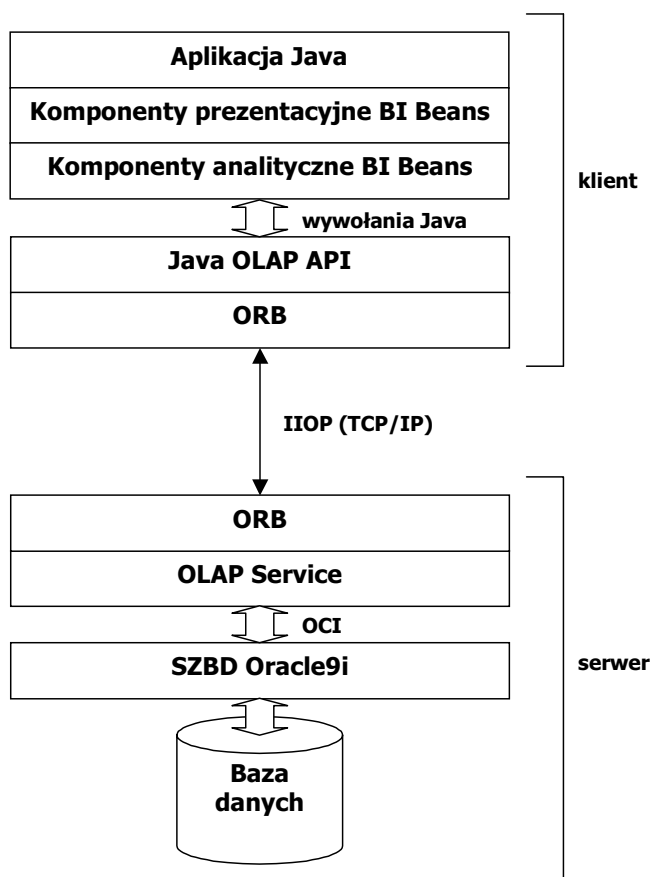
4 Komponenty Business Intelligence Beans

Interfejs Java OLAP API umożliwia programiście realizację operacji przetwarzania danych na bardzo niskim poziomie. Pozwala to konstruować aplikacje o bardzo bogatej funkcjonalności, lecz wymaga od programisty dużego wkładu pracy i bardzo dobrej znajomości klas i metod Java OLAP API.

Proces tworzenia aplikacji Java dla magazynów danych może ulec znacznemu skróceniu i uproszczeniu dzięki zastosowaniu gotowej biblioteki komponentów Business Intelligence Beans (BI Beans). BI Beans to komponenty programowe zgodne ze specyfikacją JavaBeans,

zaprojektowane jako składniki wizualnych aplikacji analitycznych opartych na Oracle9i OLAP. Wewnętrznie, komponenty BI Beans oparte są na interfejsie Java OLAP API, korzystając z jego bogatej funkcjonalności, a jednocześnie ukrywając jego skomplikowanie. Użycie komponentów BI Beans zwalnia programistę Java z konieczności rozwiązania dwóch problemów: komunikacji z magazynem danych oraz graficznej prezentacji wyników zapytań.

Komponenty BI Beans możemy podzielić na dwie główne grupy: komponenty prezentacyjne i komponenty analityczne. Komponenty prezentacyjne odpowiadają za wyświetlanie wyników zapytań do magazynu danych. Komponenty analityczne służą do izolacji mechanizmów prezentacji danych od interfejsu Java OLAP API. Żądania kierowane przez komponenty prezentacyjne do komponentów analitycznych są przekładane na stosowne wywołania interfejsu niższego poziomu. Ogólna architektura aplikacji analitycznej opartej na komponentach BI Beans została przedstawiona na rys. 3.

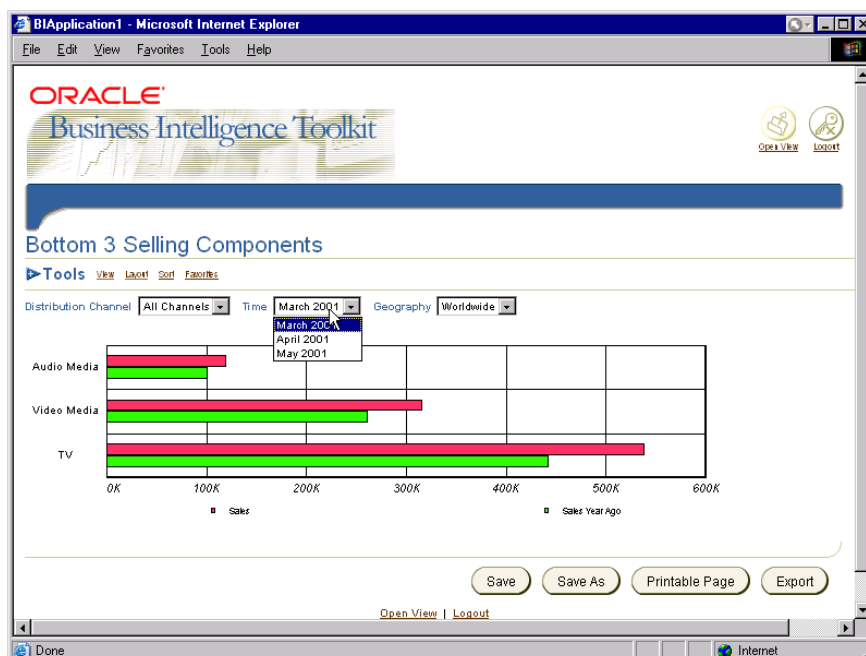


Rys. 3. Ogólna architektura aplikacji analitycznej opartej na BI Beans

Zbiór komponentów prezentacyjnych BI Beans zawiera następujące elementy: Table, Crosstab i Graph. Komponent Table służy do prezentacji danych w układzie jednowymiarowej tabeli rekordów. Komponent Crosstab reprezentuje dwuwymiarowy, tablicowy wycinek wielowymiarowej kostki danych. Komponent Graph służy do wizualizacji danych w formie dwu- lub trójwymiarowego wykresu graficznego. Wszystkie wspomniane komponenty obsługują interakcję z użytkownikiem, pozwalając na dynamiczną reorganizację sposobu wyświetlania oraz zakresu przedstawianych danych (np. operacje zwijania, rozwijania, obracania, itd.).

Komponenty prezentacyjne BI Beans mogą być wykorzystywane zarówno w aplikacjach klient-serwer (samodzielna aplikacja Java, applet Java), jak i w internetowych aplikacjach wielowarstwowych (serwlet Java, aplikacja JSP). W przypadku aplikacji wielowarstwowych,

komponenty prezentacyjne odpowiadają za generowanie dynamicznych dokumentów HTML, przesyłanych następnie na ekran przeglądarki użytkownika. Przykład działania takiej aplikacji został przedstawiony na rysunku 4.



Rys. 4. Internetowa aplikacja analityczna wykorzystująca komponent Graph

Komponenty BI Beans mogą przechowywać szczegóły swojej konfiguracji specjalnym repozytorium umieszczonym w bazie danych, nazywanym Katalogiem BI Beans. Dane konfiguracyjne komponentów BI Beans zapisywane są tam w formacie XML. Katalog BI Beans umożliwia współdzielenie obiektów analitycznych (tabel, wykresów, itp.) definiowanych zarówno przez programistę, jak i przez samego użytkownika. W ten sposób, na przykład, analityczny wykres graficzny prezentujący zawężone przez użytkownika dane z magazynu danych, może być dostępny dla pozostałych użytkowników aplikacji.

5 Podsumowanie

Komponenty Business Intelligence Beans stanowią dla programisty Java interesujące rozwiązanie, umożliwiające szybką i łatwą budowę aplikacji analitycznych operujących na magazynie danych obsługiwanych przez Oracle9i. Implementacja takich aplikacji jest niezależna od wewnętrznej reprezentacji danych w bazie danych (relacyjna lub wielowymiarowa). Z kolei wykorzystanie protokołu IIOP jako mechanizmu komunikacyjnego pozwala na fizyczną separację warstwy aplikacyjnej od warstwy magazynu danych.

6 Literatura

- [1] Oracle9i OLAP Services, Concepts and Administration Guide, Oracle
- [2] Oracle9i OLAP Services, Developer's Guide to Oracle OLAP API, Oracle
- [3] Oracle9i Business Intelligence Beans, An Oracle White Paper