ZESZYTY NAUKOWE POLITECHNIKI ŁÓDZKIEJ
Nr 1078 ELEKTRYKA, z. 121 2010

RADOSŁAW ADAMUS
TOMASZ MAREK KOWALSKI
KAMIL KULIBERDA
JACEK WIŚLICKI
Wydział Elektrotechniki Elektroniki Informatyki i Automatyki
Politechniki Łódzkiej

PROJEKT ODRA – ZAŁOŻENIA I REZULTATY

Recenzent: **prof. Dominik Sankowski**

Maszynopis dostarczono: 1. 10. 2010

Celem projektu ODRA (Object Database for Rapid Application development) jest opracowanie obiektowo zorientowanego systemu zarządzania obiektową bazą danych. System ten ma udostępniać zestaw narzędzi powalających na budowanie aplikacji biznesowych w postaci aplikacji rozproszonych, webowych, systemów usług biznesowych, wirtualnych repozytoriów, sieci P2P i wiele innych. Artykuł przedstawia podstawowe założenia oraz dotychczasowe rezultaty projektu ODRA przedstawiając podstawę teoretyczną w postaci podejścia stosowego, obecną architekturę platformy oraz zestaw najważniejszych opracowywanych mechanizmów i własności systemu.

1. WPROWADZENIE

Najbardziej rozpowszechniony paradygmat budowania aplikacji zakłada, że istnieją dwie odrębne przestrzenie danych programu – przestrzeń danych ulotnych (których istnienie ograniczone jest czasem uruchomienia aplikacji) oraz przestrzeń danych trwałych (których egzystencja rozciąga się na czas dłuższy niż

pojedyncze uruchomienie aplikacji). Odrębność ta ma bezpośredni wyraz w wielu aspektach związanych ze sposobem zarządzania danymi. Można do nich zaliczyć:

- fizyczne miejsce przechowywania - dane ulotne, przechowywane są w pamięci programu a dane trwałe w bazie danych,
- zarządzanie danymi - dane ulotne zarządzane są przez mechanizmy środowiska uruchomieniowego aplikacji, natomiast za dane trwałe odpowiada system zarządzania bazą danych
- paradygmat danych – dane ulotne mogą być reprezentowane przy wykorzystaniu całego szeregu paradygmatów, które stoją u podstaw model programowania (np. strukturalny, obiektowy, funkcyjny). Model danych trwałych może również być implementowany za pomocą różnych paradygmaty, jednak praktyka pokazuje, iż dominantą w tym zakresie jest model relacyjny (ze względu na standard języka SQL) oraz coraz bardziej popularne modele danych opartych o standard XML. Powoduje to potrzebę mapowania pomiędzy modelem aplikacji a modelem danych.
- języku manipulacji danymi – dane ulotne są manipulowane przy wykorzystaniu konstrukcji języka programowania aplikacji, który posiada semantykę pozwalającą na ich tworzenie i manipulację. Manipulacja danymi trwałymi odbywa się za pomocą odrębnych języków, udostępnianych przez systemy zarządzania bazami danych i łączonych z językiem i środowiskiem programowania aplikacji za pomocą mniej lub bardziej wyspecjalizowanych interfejsów programistycznych,
- podejściu do systemu typów oraz kontroli typologicznej – do danych trwałych i danych ulotnych stosowane są różne reguły kontroli typologicznej, wykonywanej często w różnym czasie (w trakcie kompilacji, w trakcie działania),
- wielodostęp – mechanizmy zarządzania wielodostępem do danych ulotnych są inne i wymagają innych narzędzi niż mechanizmy wielodostępu dla danych trwałych.

Największym problemem związanym z odrębnością przestrzeni danych ulotnych i trwałych, jest nie tyle możliwość wyboru z pośród mniej lub bardziej ograniczonego zestawu narzędzi, ile problem niedopasowania modelu danych ulotnych oraz modelu danych trwałych w obrębie tej samej aplikacji. Powoduje to potrzebę konwersji jednego modelu na drugi, stosowania uproszczeń, zwiększania się poziomu skomplikowania aplikacji oraz narzędzi służących ich tworzeniu.

Niniejszy artykuł opisuje założenia i dotychczasowe rezultaty projektu o nazwie ODRA (Object Database for Rapid Application development). Głównym celem badań jest opracowanie nowego paradygmatu projektowania aplikacji

wykorzystujących dane trwałe i ulotne. Założeniem paradygmatu jest unifikacja sposobów przechowywania oraz przetwarzania danych.

Dalsza część artykułu zorganizowana jest następująco: sekcja druga opisuje genezę powstania projektu, sekcja trzecia szczegółowo opisuje cele projektu ODRA. Sekcja czwarta wprowadza podstawowe pojęcia związane z podejściem stosowym (SBA). Sekcja piąta opisuje architekturę systemu w ujęciu funkcjonalnym oraz przetwarzania zapytań. Sekcja szósta wprowadza model danych systemu. Sekcja siódma stanowi przegląd najistotniejszych cech platformy. Sekcja ósma opisuje dalsze możliwe drogi rozwoju prac badawczych projektu. Sekcja dziewiąta podsumowuje artykuł.

2. GENEZA PROJEKTU

Projekt ODRA został powołany do życia w Katedrze Systemów Informatycznych Polsko Japońskiej Wyższej Szkoły Technik Komputerowych w Warszawie w zespole kierowanym przez prof. Kazimierza Subietę. Został on zainicjowany w roku 2006 jako kolejna odsłona badań nad konstrukcją obiektowych systemów baz danych opartych na podejściu/architekturze stosowej (SBA – Stack Based Approach/Architecture), które autorem jest prof. Subieta [15]. Zespół Inżynierii Oprogramowania i Baz Danych działający przy Katedrze Informatyki Stosowanej PŁ już we wczesnym etapie dołączył do zespołu realizującego projekt ODRA. Oprócz pracowników PJWSTK oraz PŁ w badaniach biorą udział naukowcy z wielu innych jednostek akademickich w Polsce (m. in. Politechniki Warszawskiej, Uniwersytetu Warszawskiego, Uniwersytetu Łódzkiego, UMK w Toruniu)

Badania wykonywane w ramach projektu ODRA zostały wykorzystane w kilku krajowych i międzynarodowych projektach. Do najważniejszych z nich należą: projekt e-Gov Bus - Advanced e-Government Information Service Bus [6] oraz VIDE - VIsualize all moDel drivEn programming [1]. System ODRA stał się również podstawą opracowania system przepływu prac o rozszerzonej funkcjonalności [5].

W chwili obecnej projekt ODRA jest kontynuowany w ramach grantu badawczego „Framework do szybkiego projektowania aplikacji webowych typu „data-intensive" oparty na środowisku systemu ODRA", który otrzymał finansowanie MNiSW na lata 2010-2012 wykonywanego w Katedrze Informatyki Stosowanej Politechnik Łódzkiej.

3. CEL PROJEKTU

Głównym celem projektu ODRA jest zbudowanie narzędzia do szybkiego budowania aplikacji opartych o przetwarzanie danych, które będzie udostępniało jedno spójne środowisko programowania abstrahujące od pojęcia trwałości.

Podstawowe założenia projektu zostały sformułowane następująco:

- Aplikacje, które są wpierane przez platformę są związane z, potencjalnie dużą, ilością danych.
- Niezależnie od tego, czy dane są trwałe czy ulotne, wszystkie konstrukcje językowe są takie same. W ogólności oznacza to, że dysponujemy językiem programowania łączącym w sobie semantykę klasycznego języka programowania oraz języka zapytań dla baz danych.
- Podstawowym podziałem rodzajów danych nie jest podział na dane trwałe i ulotne ale na dane lokalne i dane współdzielone. Oznacza to, że głównym czynnikiem warunkującym działaniem mechanizmów jest wielodostęp. Programista definiując schemat danych ma możliwość określenia zakresu dostępu do danych. System ODRA umożliwia pisanie i uruchamianie przez użytkowników aplikacji, wykorzystujących dane lokalne oraz dane współdzielone. Współdzielenie danych (regulowane przez politykę dostępu) może odbywać się na poziomie aplikacji uruchomionych w obrębie rozproszonego systemu ODRA.
- System udostępnia obiektowy model danych. Podstawową zasadą jest relatywizm obiektowy oznaczający, że wszystkie elementy definiowane w systemie są obiektami. Dotyczy to samych danych przechowywanych w bazie danych jak i meta-danych (np. klasy, procedury perspektywy) przechowujących informacje o schemacie.
- System ODRA udostępnia rozszerzalny system typów oraz mechanizmy mocnej i północnej kontroli typologicznej wykorzystywanej w procesie kompilacji aplikacji i zapytań.
- System ODRA udostępnia silne mechanizmy optymalizacji zapytań oparte na przepisywaniu zapytań oraz strukturach danych i mechanizmach przetwarzania.
- System ODRA pozwala na budowanie baz danych, których zadaniem jest integracja rozproszonych i heterogenicznych zasobów do wspólnego modelu obiektowego i które mogą być potem wykorzystywane przez aplikacje korzystające ze zintegrowanych danych.

Spełnienie tych postulatów wymaga przede wszystkim oparcia się na odpowiedniego podejścia do semantyki języka zapytań oraz samej architektury systemu zarządzania bazą danych.

4. PODEJŚCIE STOSOWE

Podejście stosowe (SBA – Stack Based Architecture) [3] jest metodologią dotyczącą budowy obiektowo zorientowanych języków zapytań i języków programowania [15]. W przeciwieństwie do klasycznego podejścia rozdzielającego języki zapytań i programowania, podejście stosowe zakłada, że

nie ma żadnych podstaw do tego typu podziału. Język zapytań i język programowania powinien być opisywany przez jedną, zunifikowaną teorię, definiująca semantykę obydwu aspektów języka. Podejście stosowe definiuje semantykę takiego języka w postaci abstrakcyjnej implementacji. Oznacza to, że definicja języka oparta jest na operacyjnej semantyce abstrakcyjnej maszyny, której zadaniem jest wykonanie zapytania czy też programu. Zasada działania takiej maszyny opisana jest za pomocą zasady działania operatorów języka na zdefiniowanych uprzednio strukturach danych. Podejście stosowe wprowadza trzy podstawowe elementy takiego środowiska wykonawczego: skład obiektów, stos środowiskowy oraz stos rezultatów. Struktura taka pozwala na precyzyjne opisanie działania wszystkich operatorów, włączając w to również operatory znane z języków zapytań (np. selekcja, projekcja, złączenie).
Elementem podejścia stosowego jest również systematyzacja kwestii modeli obiektowych. Założeniem tej idei jest sprowadzenie wielu niekompatybilnych notacji do wspólnego mianownika i opracowanie rozszerzalnej rodziny modeli składów obejmującej popularne modele i dającej możliwość opracowywania nowych w ramach tej samej ramy architektonicznej i spójnej notacji. SBA wprowadza rodzinę abstrakcyjnych modeli składu oznaczanych jako ASn (Abstract Store), gdzie n reprezentuje poziom składu. Najprostszy model AS0 pozwala na opisane modelu relacyjnego, wraz z relacjami zagnieżdżonymi czy też modelu XML. AS0 wprowadza hierarchię obiektów bez ograniczeń na poziom zagnieżdżenia, oraz umożliwia tworzenie powiązań pointerowych pomiędzy obiektami. Kolejny model AS1, jest rozszerzeniem modelu AS0 o statyczne dziedziczenie. Model AS2 rozszerza AS0 o dziedziczenie dynamiczne. Natomiast model AS3 rozszerza modele AS1 i AS2 o koncepcję hermetyzacji.

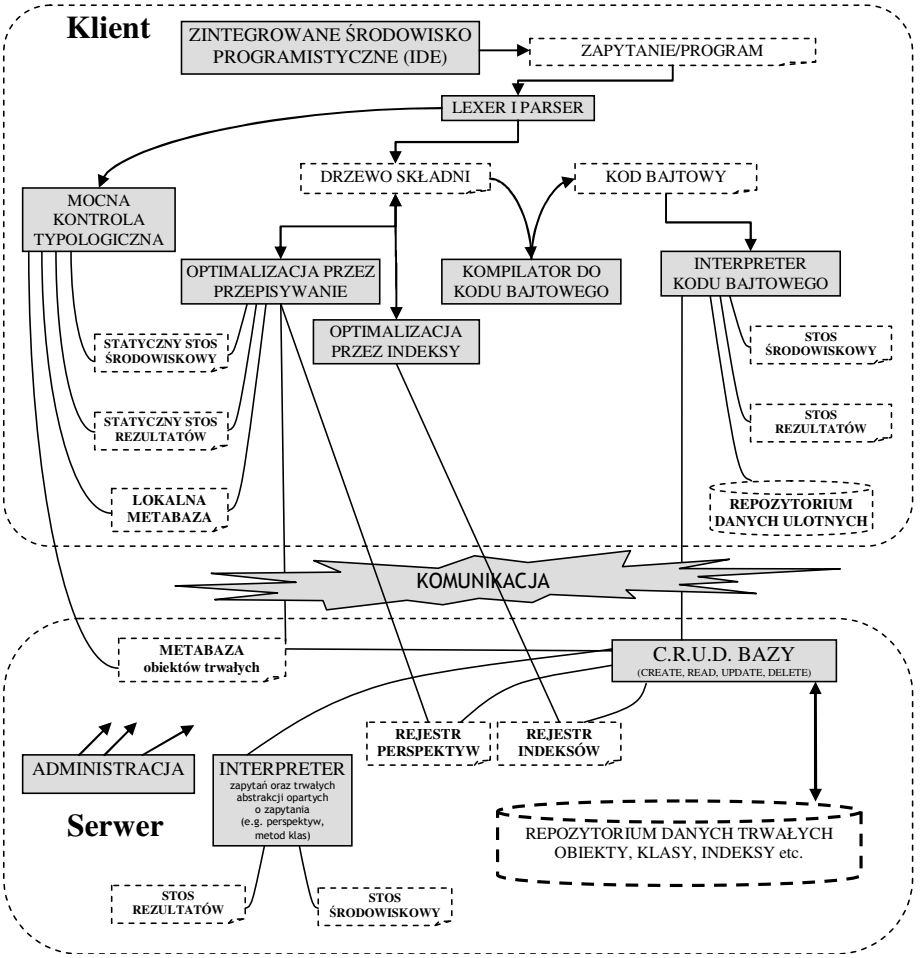
5. ARCHITEKTURA SYSTEMU

Punktów widzenia na architekturę systemów jest wiele. Na potrzeby artykułu przyjmiemy dwa takie punkty widzenia. Pierwszy z nich przedstawia architekturę systemu ODRA z punktu widzenia jej wewnętrznej logicznej organizacji. Pozwoli to na lepsze zrozumienie funkcjonalności systemu oraz możliwych sposobów jego wykorzystania. Drugi punkt widzenia to architektura przetwarzania zapytań, która przybliży schemat działania systemu w odpowiedzi na konkretne zlecenie przetwarzania danych.

5.1. Logiczna organizacja systemu

Rys. 1 przedstawia punkt widzenia na architekturę systemu uwzględniający struktury danych oraz moduły programów. Architektura uwzględnia również podział komponentów pomiędzy procesy klienta oraz serwera. Należy zwrócić uwagę iż obecnie każda instancja systemu może działać jako klient i jako serwer.

Każdy serwer obsługuje połączenia z wieloma klientami.



Rys. 1. Architektura systemu ODRA [14]

5.2. Architektura przetwarzania zapytań

Ze względu na kompozycyjność języka SBQL, pojęcie zapytania w systemie ODRA odnosi się praktycznie do wszystkich wyrażeń spotykanych w językach programowania. Zapytaniem jest proste wyrażenie składające się z np. z liczby ciągu znaków czy nazwy. Bardziej skomplikowane zapytania budowane są poprzez łączenie pod-zapytań za pomocą operatorów. Ważną cechą języka SBQL jest iż, poza znanymi z języków programowania operatorami, wprowadza on również zestaw tzw. operatorów nie-algebraicznych, które umożliwiają wykonywanie operacji na kolekcjach. To właśnie ta grupa operatorów

wprowadza na poziom języka programowania, znane z języków zapytań, konstrukcje takie jak selekcja projekcja, złączenie czy kwantyfikatory. Co bardzo istotne, semantyka tych operatorów została zdefiniowana za pomocą tych samych elementów co pozostałe elementy języka (z technicznego punktu widzenia, podstawowym wyróżnikiem operatorów nie-algebraicznych jest wpływ na środowisko przetwarzania reprezentowane przez stos środowisk [15]). Umożliwia to wykorzystanie pełnej mocy języka programowania oraz języka zapytań (włącznie z operacjami aktualizacji) w jednym, homogenicznym środowisku przetwarzania. Cecha ta jest inherentna dla semantyki podejścia stosowego i opartego na nim języka SBQL. Wszystko to osiągnięte zostało na bazie bardzo prostego aparatu teoretycznego umożliwiającego precyzyjne zdefiniowanie semantyki języka. Stawia to całą architekturę SBA w opozycji do popularnych języków i środowisk programistycznych wykorzystywanych obecnie, gdzie cechy języka zapytań oraz pełnego języka programowania są rozłączne semantycznie a próby połączenia ich w jednym środowisku opierają się bądź to na skomplikowanych teoretycznie zasadach i niejednorodnej składni bądź na ograniczaniu zestawu dostępnych operatorów.

Proces przetwarzania zapytania w systemie ODRA składa się z kilku oddzielnych etapów:

1. Rozbiór gramatyczny (parsowanie) tekstu zapytania – na podstawie tekstu zapytania budowane jest abstrakcyjne drzewo składni (AST).
2. Statyczna ewaluacja zapytania– drzewo składni zapytania poddawane jest kontroli typologicznej mającej na celu wykrycie błędów typologicznych zapytania. Moduł statycznej ewaluacji symuluje wykonanie zapytania na zbudowanym, z wykorzystaniem meta-danych dostępnych w systemie, statycznym środowisku. Mechanizm kontroli typologicznej zaimplementowanej w systemie ODRA można określić jako pół-mocny. Oznacza to, że wszędzie tak gdzie kompilator jest w stanie określić dokładnie typy w kontrolowanej części zapytania, tam stosowana jest kontrola mocna. Natomiast wszędzie tam, gdzie typ nie jest dokładnie określony, elementy kontroli typologicznej przenoszone są do czasu wykonania. Takie podejście jest szczególnie istotne w klasie języków zapytań, ponieważ elementem typu, istotną z punktu widzenia przetwarzania, jest informacja nt. liczności rezultatu danego pod-zapytania. Przeniesienie kontroli typologicznej pozwala na wykonanie szerszej klasy zapytań kosztem zmniejszenia szczelności systemu typów. Należy jednak zwrócić uwagę, że ważna jest tutaj budowa mechanizmów kontroli czasu wykonania, tak aby maksymalnie zminimalizować skutki błędów typologicznych wykrytych w czasie wykonania. Osiągniecie tego celu wymaga wprowadzenia modyfikacji w drzewie składni uzupełniających informację dla potrzeb modułu wykonawczego.

3. Optymalizacja – drzewo składni, powstałe w rezultacie statycznej ewaluacji zostaje poddane optymalizacji przez konfigurowalny zestaw optymalizatorów przepisujących. Termin przepisujący oznacza w tym kontekście, iż rezultatem optymalizacji jest zmodyfikowane (przepisane drzewo zapytania).
4. Generacja kodu – zoptymalizowane drzewo składni zapytania jest przekazywane do generatora kodu bajtowego, który zamienia strukturę drzewiastą na liniowy kod bajtowy. System ODRA posiada własny kod bajtowy o nazwie Juliet. Wszystkie programy oraz zapytania są kompilowane do tej postaci.
5. Interpretacja – ostatnim etapem przetwarzania jest wykonanie kodu bajtowego zapytania przez interpreter.

6. MODEL DANYCH

Obiektowy model danych systemu ODRA jest zbliżony do modelu języka UML. Pozwala to na transformację diagramów klas języka UML do schematu danych w systemie. Poziom odwzorowania modelu UML ograniczony jest tym, że UML jest językiem modelowania i nie wszystkie jego cechy dają się bezpośrednio zmapować ma model danych. Model systemu jest, poza kilkoma mniej znaczącymi cechami, zgodny z modelem języka XML.

6.1. Obiekt

Podstawowym pojęciem modelu danych systemu ODRA jest obiekt. Każdy obiekt ma zewnętrzną nazwę, która pozwala na dowiązanie się do niego z poziomu kodu źródłowego programu. Nazwa obiektu, która nie musi być unikatowa, posiada określone znaczenie z punktu widzenia dziedziny biznesowej aplikacji. Obiekt posiada również wewnętrzny identyfikator, który jest niejawnie wykorzystywany jako referencja obiektu. Identyfikator obiektu jest unikatowy w obrębie danego środowiska obiektu. Identyfikatory te nie są jawnie wykorzystywane w aplikacjach i nie mają żadnego znaczenia z punktu widzenia dziedziny biznesowej aplikacji.

Brak ograniczenia co do unikatowości nazwy obiektu w danym środowisku pozwala na bezpośrednie wprowadzenie pojęcia kolekcji jako zestawu obiektów o tej samej nazwie (i w większości przypadków również typie) [4]. Identyczność nazwy jest jedynym wyznacznikiem kolekcji. Z tego też powodu zmiana kolekcji wiąże się ze zmianą nazwy obiektu (w przypadku modelu z klasami i dziedziczeniem kwestia ta jest trochę bardziej skomplikowana). Takie podejście do pojęcia kolekcji powoduje, że nie ma najmniejszego problemu z zagnieżdżaniem kolekcji w obiekcie.

6.2. Wiązanie obiektów

Obiekty w systemie ODRA mogą być ze sobą wiązane za pomocą specjalnego typu obiektu zwanego obiektem pointerowym (albo po prostu pointerem). Wartością takiego obiektu jest referencja innego obiektu w składzie. ODRA umożliwia definiowanie bliźniaczych obiektów pointerowych pozwalających na implementację dwukierunkowych asocjacji binarnych znanych z modelu UML.

6.3. Organizacja obiektów

Podstawową jednostką organizacji obiektów w systemie ODRA jest moduł. Stanowi on niezależny komponent systemu grupujący obiekty reprezentujące dane oraz skompilowane programy oraz przechowujący ich schemat w postaci tzw. metabazy modułu.

6.4. Klasy obiektów

Klasa obiektów jest abstrakcją programistyczną definiująca właściwości obiektów będących instancjami klasy. W językach programowania klasa jest często elementem drugiej kategorii programistycznej – występuje tylko na poziomie kodu źródłowego programu. W niektórych językach i środowiskach klasa może być reprezentowana w środowisku wykonawczym w postaci specjalnego obiektu wykorzystywanego w mechanizmach refleksji. W systemie ODRA klasa jest obiektem przechowywanym w bazie danych i przechowującym obiekty reprezentujące między innymi .skompilowany kod metod. Dodatkowo w obszarze meta danych przechowywana jest meta-klasa opisująca typ powiązany z klasą (definiowany poprzez zestaw typów właściwości klasy).

6.5. Dziedziczenie i polimorfizm

Podobnie jak w modelu języka UML czy innych popularnych modeli obiektowych definiowanie klasy obiektów może uwzględniać powtórne wykorzystanie istniejącej definicji poprzez mechanizm dziedziczenia. Ze względu na reprezentację klas w systemie ODRA w postaci obiektów przechowywanych w bazie danych samo dziedziczenie pomiędzy klasami.

7. WŁASNOŚCI PROTOTYPOWEGO SYSTEMU ODRA

Niniejsza sekcja opisuje najważniejsze własności prototypowej implementacji systemu ODRA.

7.1. Skład danych

Mechanizm składu danych odpowiada za organizację podstawowych struktur danych składowanych w bazie danych. Zarządzane przez skład struktury znajdują się w pamięci, która historycznie może być podzielona na ulotną (pamięć operacyjna) oraz trwałą (pamięć dyskowa). Ulotny skład danych wykorzystywany jest w klasycznych językach programowania i nosi najczęściej miano sterty (ang. heap). Trwały skład danych związany jest z bazami danych. W systemie ODRA mechanizmy składu danych abstrahują od pojęcia trwałości i mogą zostać zastosowane zarówno do pamięci operacyjne jak i pamięci dyskowej.
Implementacja składu danych w systemie ODRA podzielona została na cztery części [11]. Najbardziej podstawową jest warstwa fizyczna składu zarządzająca plikami lub blokami pamięci. Kolejną warstwą jest warstwa zarządzania danymi, implementująca mechanizmy zarządzania wolną przestrzenią w ramach bloków danych. Jej zadaniem jest udostępnienie trwałej lub ulotnej ciągłej przestrzeni przechowującej i organizujące dane w bazie. W tym celu warstwa zarządzania danymi może grupować pliki lub bloki pamięci w jedną ciągłą przestrzeń umożliwiającą adresowanie. Bazująca na tej implementacji warstwa obiektów składu danych implementuje podstawową funkcjonalność zgodną z modelami danych SBA znanymi pod nazwami AS0, AS1. Warstwa obiektów składu danych jest z kolei podstawą warstwy obiektów bazy danych. Jej zadaniem jest wyrażenie wszystkich bytów bazy danych związanych z przyjętym modelem danych oraz architekturą mechanizmów bazy danych. Mogą to być procedury, funkcje, metody, perspektywy, wirtualne, atrybuty, ograniczenia, bloki obsługi zdarzeń, reguły, procesy, wątki i inne elementy proceduralne, moduły, elementy metabazy, indeksy, itd. Większość mechanizmów Odry (kompilator, optymalizatory, interpreter SBQL, itd.) korzysta tylko z warstwy obiektów.

7.2. Wirtualne obiektowe perspektywy

W systemach bazodanowych perspektywa (widok) jest zdefiniowanym obrazem danych. Mechanizm perspektyw daje możliwość odkreślania sposobów dostępów do danych niezależnie od ich fizycznej struktury. W ten sposób programista ma możliwość zdefiniowania „wirtualnych” danych, których sposób przetwarzanie nie powinno odróżniać się od sposobu wykorzystywanego przy przetwarzaniu danych rzeczywistych. Z technicznego punktu widzenia mapowanie takie wykonywane jest przez zapytanie a perspektywa jest zapisanym zapytaniem. Schemat danych udostępnianych poprzez widok jest schematem rezultatu tego zapytania. Odwołanie się do perspektywy oznacza wykonanie zapytania. Przezroczystość mechanizmu operowania na perspektywach jest często ograniczona dla operacji aktualizacyjnych. W języku SQL nie ma mechanizmów zdefiniowania semantyki aktualizacji perspektywy co

powoduje, że bardziej skomplikowane widoki, które na przykład w swojej definicji wykorzystują funkcje agregujące nie mogą być aktualizowane. W celach optymalizacyjnych stosuje się zabieg materializacji „wirtualnych" danych. Takie zmaterializowane perspektywy przechowywane są w bazie danych w postaci zapisanego rezultatu zapytania.

System ODRA posiada implementację obiektowych perspektyw [2][9] nie posiadającą żadnych ograniczeń w kwestii możliwości aktualizacyjnych. Osiągnięte to zostało poprzez rozszerzenie pojęcia definicji perspektywy. Definicja ta składa się z dwóch części. Pierwsza część to zapytanie konstruujące „wirtualne obiekty" dostępne poprzez widok. Informacje w nim zawarte powinny umożliwiać aktualizację danych. Rezultatem wykonania tego zapytania jest kolekcja „wirtualnych identyfikatorów" symulujących rzeczywiste identyfikatory obiektów składu. Druga część definicji to zestaw przedefiniowanych operatorów działających na „wirtualnych obiektach". Dzięki temu programista może wyrazić biznesową semantykę aktualizacji danych wirtualnych lub zabronić aktualizacji poprzez nie umieszczanie definicji wybranej operacji. Na podstawowy zestaw operatorów, które definiuje programista składają się: operator dereferencji określający rezultat zapytania wykonywanego na wirtualnym obiekcie, operator aktualizacji definiujący operacje wykonywane w momencie aktualizacji wirtualnego obiektu, operator usuwania definiujący semantykę usuwania wirtualnego obiektu oraz operator tworzenia nowego obiektu wirtualnego. Na definicję perspektywy mogą składać się również definicję pod-perspektyw reprezentujących wirtualne pod-obiekty obiektu nadrzędnego.

Mechanizm perspektyw stanowi podstawę wielu innych funkcjonalności systemu ODRA. Wykorzystywany jest między innymi do budowania wirtualnych danych reprezentujących dane relacyjne dostępne za pomocą obiektowej osłony [18].

7.3. Statyczna pół-mocna kontrola typologiczna

Kontrola typologiczna w systemie ODRA opiera się na nowej teorii dotyczącej mocnej kontroli typologicznej [16]. Wyróżnia ona wewnętrzny i zewnętrzny system typów. Wewnętrzny system typów odzwierciedla zachowanie się mechanizmu kontroli typologicznej. Zewnętrzny system typów jest wykorzystywany przez programistę. Mechanizm statycznej kontroli typologicznej symuluje w trakcie kompilacji, działanie mechanizmów czasu wykonania. Do symulacji wykorzystuje statyczne struktury danych: statyczny stos środowiskowy, statyczny stos rezultatów oraz metabazę. Przetwarzanie na stosach statycznych oparte jest o tzw. sygnatury – statyczne odpowiedniki rezultatów zapytań z czasu wykonania, przechowujące informacje typologiczne. Oprócz informacji typologicznej sygnatury są nośnikiem dodatkowych atrybutów przenoszących informacje takie jak liczność, rodzaj kolekcji, nazwę typu, zmienność.

7.4. Optymalizacja zapytań

System ODRA wyposażony został w zestaw optymalizatorów przepisujących. Zaimplementowane zostały następujące metody optymalizacji: metoda niezależnych podzapytań [13], usuwanie martwych podzapytań [13], usuwanie nazw pomocniczych, przepisywanie perspektyw, optymalizacja przy użyciu indeksów [8], metoda optymalizacji zapytań rozdzielnych względem operatora unii zbiorów. Wybór oraz kolejność wykonywania poszczególnych optymalizacji na drzewie składni jest konfigurowalna. Umożliwia zdefiniowanie optymalnej kolejności ze względu na konkretne zastosowanie oraz własności przechowywanych danych.

7.5. Przetwarzanie rozproszone

Jednym z podstawowych celów projektu ODRA jest opracowanie systemu pozwalającego na integrację rozproszonych danych. Koncepcja na jakiej opiera się rozwiązanie zaimplementowane w systemie ODRA opiera się na idei federacyjnych baz danych czy też systemów typu grid bazodanowy. Podejścia te zakładają istnienie wspólnego modelu danych oraz, opartego na nim, globalnego schematu, który jest podstawą do opracowywania aplikacji klienckich w stosunku do gridu. Od strony konstrukcji gridu systemy udostępniające swoje zasoby poprzez mechanizmy gridowe muszą posiadać narzędzia pozwalające na dostosowanie własnego modelu i schematu danych do modelu i schematu gridu.
Dostosowanie schematu oznacza dokonanie takich przekształceń aby dostosować lokalny schemat danych do schematu globalnego. Mechanizm ten może wymagać istnienia wyspecjalizowanych modułów zwanych mediatorami, których możliwości powinny dawać maksymalnie szeroki zestaw możliwych przekształceń oraz, w razie potrzeby, udostępniać mechanizmy aktualizacji danych lokalnych poprzez dane globalne. W systemie ODRA mechanizm mediatorów jest realizowany z wykorzystaniem aktualizowanych obiektowych perspektyw, spełniających powyższe postulaty [10].
Dostosowanie modelu jest bardziej skomplikowane i wymaga wprowadzenia poziomu osłon (np. obiektowo-relacyjnych), co znaczenie komplikuje systemy i stosowane powinno być tylko w przypadku potrzeby udostępnienia poprzez grid danych spadkowych z istniejących źródeł [18].

7.6. Osłony

Dla potrzeb integracji danych spadkowych, które najczęściej przechowywane są w relacyjnych bazach danych, system ODRA wyposażony został w mechanizm obiektowych osłon do relacyjnych baz danych, która umożliwiają wykorzystanie danych relacyjnych w obiektowej bazie danych [18]. Dzięki wykorzystaniu funkcjonalności wirtualnych obiektowych perspektyw możliwa

jest dwukierunkowe mapowanie. Z jednej strony dane relacyjne są mapowane na wirtualne obiekty dostępne z poziomu obiektowych aplikacji działających na systemie ODRA. Z drugiej strony operacje aktualizacyjne wykonywane na wirtualnych danych są mapowane na operacje wykonywane na danych relacyjnych. Bardzo istotną cechą osłon jest ich zdolność do wykorzystania optymalizatora relacyjnego. Specjalny optymalizator przepisujący powiązany z osłoną wykorzystuje informacje o optymalizacjach dostępnych na poziomie osłanianego systemu relacyjnego i dokonuje przepisania zapytania w taki sposób, aby system relacyjny miał możliwość zoptymalizowania zapytania relacyjnego [17].

7.7. Indeksy

Indeksy są pomocniczymi strukturami przechowywanymi w bazie danych. Administrator może zarządzać zestawem indeksów poprzez dodawanie lub usuwanie indeksów, w zależności od potrzeb. W systemie ODRA zaimplementowane zostały indeksy bazujące na technice Liniowego Hashingu [7][8]. Implementacja wspiera przezroczystą optymalizację realizowaną przez dedykowany moduł optymalizatora przepisującego. Ważną cechą tego typu indeksów jest możliwość wykorzystania ich nie tylko dla scentralizowanego systemu, ale również dla środowisk rozproszonych.

7.8. Komunikacja z systemem

Komunikacja z systemem ODRA w obecnej implementacji możliwa jest za pomocą: dedykowanego protokołu (na którym opiera się implementacja klienta wyposażonego w interfejs linii poleceń oraz graficzne środowisko programistyczne), usługi sieciowej udostępniającej interfejs dla wykonywania zapytań, oraz łączy do języka Java oraz platformy NET [14].

7.9. Usługi sieciowe

System ODRA posiada własną implementację usług sieciowych (ang. Web Services) [14] umożliwiającą eksponowanie funkcji oraz klas w postaci usług sieciowych. Dodatkowo domyślna usługa sieciowa pozwala na zadawanie zapytań. Osobną funkcjonalnością jest możliwość konsumowania zewnętrznych usług sieciowych i wykorzystania ich w lokalnych programach.

8. DALSZY ROZWÓJ SYSTEMU

Prototypowa implementacja udowodniła, iż podejście (oparte na SBA), jakie zastosowano ma duży potencjał semantyczny pozwalający na realizacje cech, jakich popularne systemy baz danych nie posiadają lub wprowadzają w

niepełnym zakresie a ich wykorzystanie obwarowane jest licznymi ograniczeniami. Sam prototyp, ze względu na badawczy charakter prac nie reprezentuje odpowiednio dojrzałej architektury (co jest cechą budowy prototypów), która pozwalałaby na potraktowanie go jako podstawy do implementacji platformy produkcyjnej mogącej stać się narzędziem biznesowym. Dlatego ewentualne wdrożenie produkcyjne idei opracowywanych w projekcie ODRA wymaga wysiłku inżynierskiego w zakresie produkcji dojrzałego oprogramowania. Takie prace są podejmowane.

Z drugiej strony rezultaty projektu ODRA otworzyły wiele potencjalnych obszarów badawczych. Należą do nich zagadnienia rozszerzania semantyki języka SBQL lub ograniczania jej dla potrzeb określonych dziedzin zastosowania (np. generowanie aplikacji z interfejsem webowym, systemy przepływu prac, systemy kontroli wersji). Innymi obszarami wymagającymi badań są aspekty konstrukcji składu obiektów uwzględniających bardziej złożoną semantykę (dwukierunkowe asocjacje, dynamiczne role, itp.).

9. PODSUMOWANIE

Projekt ODRA i jego rezultaty pokazują, że takie obszary badawcze jak obiektowe bazy danych mogą być bardzo atrakcyjne z punktu widzenia naukowego. Jest to zaprzeczeniem poglądów spotykanych niekiedy, że nie warto inwestować wysiłku naukowego w takie obszary skoro istniejące rozwiązania się sprawdzają a do ewentualnych ich ograniczeń wszyscy się już przyzwyczaili i opracowali mechanizmy pozwalające na ich omijanie. Historia rozwoju takich korporacji jak Google pokazuje, że opracowanie i napisanie od podstaw tak popularnego i uznanego narzędzia jak przeglądarka internetowa, czy też całego systemu operacyjnego nie jest niczym absurdalnym. Oczywiście rozwój Inżynierii Oprogramowania najczęściej polega na ewolucyjnym dodawaniu kolejnych pięter do istniejących budowli. Jednak istnieje również potrzeba budowania nowych fundamentów, na których być może, ktoś inny będzie budował piętra.

LITERATURA

[1] **Adamus R., Falda G., Habela P., Kaczmarski K., Stencel K., Subieta K.**: Project VIDE: Challenges of Executable Modelling of Business Applications. Genie Logiciel., 2008, vol., no. 85, p. 53-56, ISSN: 0295-6322.

[2] **Adamus R., Kaczmarski K., Stencel K., Subieta K.**: SBQL Object Views - Unlimited Mapping and Updatability., p. 119-141. ICOODB Proceedings of the First International Conference on Object Databases.. Niemcy, 13-14.03.2008. Berlin: 2008, ISBN: 978-80-7399-4.

[3] **Adamus R., Habela P., Kaczmarski K., Letner M., Stencel K., Subieta K.**: Stack-

Based Architecture and Stack -Based Query Language., p. 77-95. ICOODB Proceedings of the First International Conference on Object Databases.. Niemcy, 13-14.03.2008. Berlin: 2008, ISBN: 978-80-7399-4.

[4] **Adamus R., Kowalski T., Subieta K., Wiślicki J., Stencel K., Letner M, Habela P., Daczkowski M., Pieciurkiewicz T., Trzaska M., Wardziak T.**: Overview of the Project ODRA., p. 179-197. ICOODB Proceedings of the First International Conference on Object Databases.. Niemcy, 13-14.03.2008. Berlin: 2008, ISBN: 978-80-7399-4.

[5] **Dąbrowski M., Drabik M., Habela P., Subieta K.**: Object-Oriented Declarative Workflow Management System. Editors of the Institute of Computer Science, Polish Academy of Sciences, ISBN 978-83-922508-3-8, 2009, 176 pages.

[6] e-Gov Bus: http://www.rodan.pl/web/guest/projekty_europejskie/egov_bus

[7] **Kowalski T., Wiślicki J., Kuliberda K., Adamus R., Subieta K.**: Optimization by Indices in ODRA., p. 97-117. ICOODB Proceedings of the First International Conference on Object Databases.. Niemcy, 13-14.03.2008. Berlin: 2008, ISBN: 978-80-7399-4.

[8] **Kowalski T.**: Przezroczyste indeksowanie w rozproszonych obiektowych bazach danych. (Transparent indexing in distributed object-oriented databases). Katedra Informatyki Stosowanej PŁ 2009, 181 s., rozprawa doktorska.

[9] **Kozankiewicz H.**: Updateable Object Views, IPI PAN, 2005, rozprawa doktorska.

[10] **Kozankiewicz H., Leszczylowski, J., Subieta K.**: Implementing Mediators through Virtual Updateable Views. In : Engineering Federated Information Systems, Proceedings of the 5th Workshop EFIS 2003, Coventry, UK, pp.52-62 (July 17-18 2003).

[11] **Kuliberda K., Adamus R., Wiślicki J., Kaczmarski K., Kowalski T., Subieta K.**: Autonomous Layer for Data Integration in a Virtual Repository. In : OTM 2006, International Conference on Grid computing, high-performAnce and and Distributed Applications (GADA'06), Springer 2006 LNCS 4276, Montpellier, France, pp.1290-1304 (October 29 - November 3, 2006).

[12] **Lentner M.**: Integracja danych i aplikacji przy użyciu wirtualnych repozytoriów, PJWSTK, 2008, rozprawa doktorska, 190 s.

[13] **Płodzień J.**: Optimization Methods in Object Query Languages, IPI PAN, 2001, rozprawa doktorska.

[14] **Subieta K.** i zespół projektu ODRA: Dokumentacja systemu ODRA, http://sbql.pl/various/ODRA/ODRA_manual.html

[15] **Subieta K.**: Teoria i konstrukcja obiektowych języków zapytań. Wydawnictwo PJWSTK, Warszawa 2004, ISBN 83-89244-29-2, 522 s.

[16] **Stencel K.**: Północna kontrola typów w językach programowania baz danych. Wydawnictwo PJWSTK, Warszawa 2006, ISBN 83-89244-50-0.

[17] **Wiślicki J., Kuliberda K., Kowalski T., Adamus R., Subieta K.**: Implementation and Testing of SBQL Object-Ralational Wrapper Supporting Query Optimisation., p. 39-57. Ed. Paterson James, Edlich Stefan. ICOODB Proceedings of the First International Conference on Object Databases. Niemcy, 13-14.03.2008. Berlin: 2008, ISBN: 978-80-7399-4.

[18] **Wiślicki J.**: Obiektowa osłona do relacyjnych baz danych z uwzględnieniem optymalizacji zapytań. Łódź: Katedra Informatyki Stosowanej PŁ 2008, 235 s., rozprawa doktorska.

ODRA PROJECT – ASSUMPTIONS AND RESULTS

Abstract

The aim of the Project ODRA (Object Database for Rapid Application development) is to develop an object oriented database management system. The system should include a set of tools simplifying the development process of various types of business applications. The types include distributed application, web applications, business services platforms, virtual repositories, P2P based applications and many others. The paper describes the project theoretical background and main assumptions of the research. Finally the results so far of the project – the prototype system ODRA - are presented. The description focus on the system architecture and the set of its main features and facilities.

Politechnika Łódzka
Katedra Informatyki Stosowanej