



POLITECHNIKA ŁÓDZKA  
WYDZIAŁ ELEKTROTECHNIKI, ELEKTRONIKI,  
INFORMATYKI I AUTOMATYKI  
KATEDRA INFORMATYKI STOSOWANEJ



**mgr inż. Jacek Wiślicki**

autoreferat do rozprawy doktorskiej

**Obiektowa osłona  
do relacyjnych baz danych  
z uwzględnieniem optymalizacji  
zapytań**

*(An object-oriented wrapper  
to relational databases  
with query optimisation)*

**promotor:  
prof. dr hab. inż. Kazimierz Subieta**

Łódź 2007

## **Spis treści**

WSTĘP .....	3
CEL I TEZY PRACY .....	4
STAN WIEDZY I PRACE POKREWNE.....	6
KONCEPCJA INTEGRACJI OBIEKTOWO-RELACYJNEJ .....	8
Wirtualne repozytorium .....	8
Przetwarzanie zapytań i metody optymalizacyjne .....	10
Podejście naiwne a optymalizacja.....	11
Przykład koncepcyjny .....	12
PODSUMOWANIE.....	17
BIBLIOGRAFIA .....	18

## Wstęp

Liczba dostępnych obecnie źródeł danych jest ogromna. Wiele z nich jest dostępne poprzez Internet, jakkolwiek mogą one nie być publiczne lub oferować dostęp ograniczony jedynie do ściśle określonej grupy użytkowników. Takie zasoby charakteryzują się rozproszeniem, niejednorodnością, fragmentacją i nadmiarowością. Idea, której część została opracowana i zaimplementowana w omawianej pracy doktorskiej polega na wirtualnej integracji takich zasobów w zcentralizowaną, jednorodną, spójną i pozbawioną fragmentacji oraz nadmiarowości całość tworzącą *wirtualne repozytorium* zapewniające pewne powszechne funkcjonalności i usługi, włączając w to infrastrukturę zaufania (bezpieczeństwo, prywatność, licencjonowanie, płatności, itp.), *Web Services*, rozproszone transakcje, zarządzanie procesami (*workflow management*), itd.

Głównym celem przedstawionych prac jest integracja heterogenicznych relacyjnych baz danych (zasobów) do obiektowego systemu bazodanowego, w którym są one widziane jako czysto obiektowe modele i składy, które mogą być w przeźroczysty sposób odpytywane za pomocą obiektowego języka zapytań (innymi słowy, muszą być one nieodróżnialne od rzeczywistych zasobów obiektowych). W związku z tym schematy relacyjne muszą zostać „osłonięte” wyspecjalizowanym oprogramowaniem zdolnym do dwukierunkowej wymiany danych pomiędzy znajdującym się na samym dole SZRBD i innymi elementami systemu zlokalizowanymi w wyższych warstwach repozytorium. Ten proces musi być przeźroczysty, aby końcowy użytkownik (człowiek lub aplikacja) nie był w żaden sposób świadomy faktycznego modelu zasobu. Z drugiej strony, obiektowa baza danych komunikująca się bezpośrednio z taką osłoną musi być całkowicie nieprzejrzysta (czarna skrzynka), aby osłona nie była w żaden sposób bezpośrednio dostępna z żadnego elementu systemu znajdującego się w górnych warstwach architektury. Kolejną cechą takiej osłony jest pełna generyczność – jej działanie, niezawodność i własności wydajnościowe powinny być całkowicie uniezależnione od zasobu, który kryje. Z założenia niedozwolone są także materializacja i replikacja danych relacyjnych po stronie wirtualnego repozytorium (ani żadnego innego pośredniego modułu systemu), oczywiście poza bieżącymi wynikami zapytań, które muszą zostać zwrócone użytkownikom odpytującym system.

## Cel i tezy pracy

Prezentowana praca doktorska jest skoncentrowana na nowatorskim podejściu względem integracji heterogenicznych zasobów relacyjnych do rozproszonego obiektowego systemu bazodanowego. Zasoby te muszą być dostępne dla globalnych użytkowników poprzez globalny model obiektowy i obiektowy język zapytań, tak aby ci użytkownicy nie byli w żaden sposób świadomi faktycznego modelu i składu zasobu. Opracowany i zaimplementowany proces integracji jest całkowicie przeźroczysty i umożliwia dwukierunkową wymianę danych, tj. odpytywanie zasobu relacyjnego (pobieranie danych zgodnych z kryteriami zapytań) i aktualizację danych relacyjnych. Ponadto, znajdujące się tuż nad zasobem relacyjnym struktury obiektowe (utworzone bezpośrednio w oparciu o ten zasób) mogą zostać bezproblemowo przekształcane i filtrowane (poprzez kaskadowo nabudowane aktualizowalne perspektywy obiektowe) w taki sposób, aby odpowiadały modelowi biznesowemu i ogólnemu schematowi, którego część mają stanowić. Poza aspektami przeźroczystości, największy wysiłek został poświęcony procedurom wydajnej optymalizacji zapytań umożliwiającej działanie natywnych optymalizatorów zasobu relacyjnego. Te funkcjonalności zostały osiągnięte poprzez wyspecjalizowany moduł stanowiący *osłonę obiektowo-relacyjną*, nazywany dalej w skrócie *osłoną*.

Tezy pracy zostały zdefiniowane w następując sposób:

- 1. Spadkowe bazy relacyjne mogą zostać w przeźroczysty sposób włączone do obiektowego wirtualnego repozytorium, a znajdujące się w nich dane przetwarzane i aktualizowane przez obiektowy język zapytań – nieodróżnialnie od danych czysto obiektowych, bez konieczności materializacji i replikacji.**
- 2. Dla takiego systemu mogą zostać opracowane i zaimplementowane mechanizmy umożliwiające współdziałanie obiektowej optymalizacji wirtualnego repozytorium z natywnymi optymalizatorami zasobu relacyjnego.**

Zrealizowane w pracy doktorskiej prototypowe rozwiązanie zostało oparte o podejście stosowe do języków zapytań i baz danych (SBA, *Stack-Based Approach*), wynikające z niego język zapytań (SBQL, *Stack-Based Query Language*) oraz aktualizowalne obiektowe perspektywy, interfejs JDBC, protokół TCP/IP oraz język SQL. Implementacja została wykonana w języku Java<sup>TM</sup>. Koncepcja z jej różnymi aspektami (włącznie z wirtualnym repozytorium, którego jest częścią oraz kwestiami

fragmentacji i integracji danych) przedstawiona została w ponad 20 artykułach, np. [1, 2, 3, 4, 5, 6].

Praca doktorska została wykonana w ramach projektu eGov-Bus (*Advanced eGovernment Information Service Bus*) wspieranego przez Wspólnotę Europejską w ramach priorytetu „Information Society Technologies” Szóstego Programu Ramowego (nr kontraktu: FP6-IST-4-026727-STP).

## Stan wiedzy i prace pokrewne

Sztuka budowania obiektowych osłon do relacyjnych baz danych rozwijana jest od około 15 lat – pierwsze publikacje z tej dziedziny pojawiły się u schyłku lat osiemdziesiątych ubiegłego wieku i były poświęcone bazom federacyjnym. Niezmiennym celem podobnych technologii jest połączenie silnie osadzonej w przemyśle i podświadomości teorii relacyjnej (podstawy zostały zaprezentowane przez Coddą prawie 40 lat temu [7]) ze znacznie młodszą teorią obiektowych baz danych – określenie *obiektowego systemu bazodanowego* zostało po raz pierwszy użyte niemalże 20 lat później w słynnym „manifestie obiektowości” [8] (aczkolwiek prace zostały rozpoczęte ponad 10 lat wcześniej).

Motywacją dla budowania osłon jest redukcja technicznych i kulturowych różnic pomiędzy tradycyjnymi bazami relacyjnymi (rozdział 3 pracy), a nowymi technologiami opartymi o paradygmat obiektowy, włączając w to metodologie analizy i projektowania (np. oparte na UML), obiektowe języki programowania (np. C++, Java, C#), obiektowe rozwiązania pośredniczące (np. oparte na CORBA), obiektowo-relacyjne i czysto obiektowe bazy danych (rozdział 4 pracy). Wszelkiego rodzaju różnice i rozbieżności między światem relacyjnym i obiektowym (zarówno na poziomie modelu danych, jak i języków zapytań i języków programowania) określane są często mianem *niedopasowania impedancji* (rozdział 2.1). W ostatnich latach potrzeba budowania osłon pojawiła się także w technologiach webowych opartych na zasobach XML/RDF. Pomimo olbrzymiej presji na zastosowanie technologii związanych z XML oraz obiektowością, ludzie (szeroko rozumiany przemysł) są wciąż zadowoleni z baz relacyjnych, w związku z czym istnieje niewielkie prawdopodobieństwo, że nastąpi masowa migracja rynku w stronę innych paradygmatów składowania i przetwarzania danych – koszt oraz czas konieczne dla przeprowadzenia takiego procesu są nieprzewidywalnie duże.

Koncepcja mediatorów i osłon wykorzystana w pracy (rozdział 2.2.1) została po raz pierwszy sformułowana przez Wiederholda [9] jako zbiór wskazówek dla rozwijanych w przyszłości systemów przetwarzania informacji. Jej inspiracją stał się powszechny wzrost zapotrzebowania na informacje zainicjalizowany przez rozwój Internetu i łącz szerokopasmowych. Zjawisko zapotrzebowania było (i wciąż jest) ograniczane przez niewydajne, pofragmentowane, niejednorodne i rozproszone źródła

danych. Podstawową ideą było zapewnienie systemom decyzyjnym elastycznych infrastruktur (określonych poprzez pojęcia mediatorów i osłon) zdolnych do pobierania kompletnej informacji bez uczestnictwa człowieka. *Mediator* został zdefiniowany jako autonomiczne oprogramowanie potrafiące przetwarzać dane z podlegającego mu źródła danych zgodnie z ogólnymi wymogami systemu. Ponieważ mediatory były postrzegane jako niezależne od zasobu, były one wyposażone w *osłony* – kolejne moduły w przezroczysty sposób pośredniczące pomiędzy zasobem, a mediatorem. Innym bardzo ważnym postulatem była niedostępność mediatorów poprzez przyjazny użytkownikowi język – dla zapewnienia wydajności i niezawodności mediacji wewnętrzne procesy miały być realizowane przez język zorientowany na komunikację. Interfejs przyjazny użytkownikowi miał być wyeksponowany na zewnątrz, gdzie działają aplikacje klienckie najwyższego poziomu. Koncepcja rozproszonej mediacji doczekała się szeregu implementacji, z których najważniejsze to Pegasus (1993) [10], Amos (1994) [11] i Amos II (rozwijany od roku 2002) [12] oraz DISCO (1997) [13] (ich krótka charakterystyka została także zawarta w rozdziale 2.2.1).

Istnieje także szereg odmiennych technologii mających na celu uzyskanie obiektowego dostępu do danych relacyjnych i przetwarzanie tych danych w obiektowy sposób (rozdziały od 2.2.2 do 2.2.5). Do tego rodzaju rozwiązań należą ORM/DAO, perspektywy XML kryjące dane relacyjne oraz wykorzystanie RDF. Ich użycie w pracach nad osłoną nie było jednak brane pod uwagę, ponieważ założenia i funkcjonalności nie są zgodne z celami pracy.

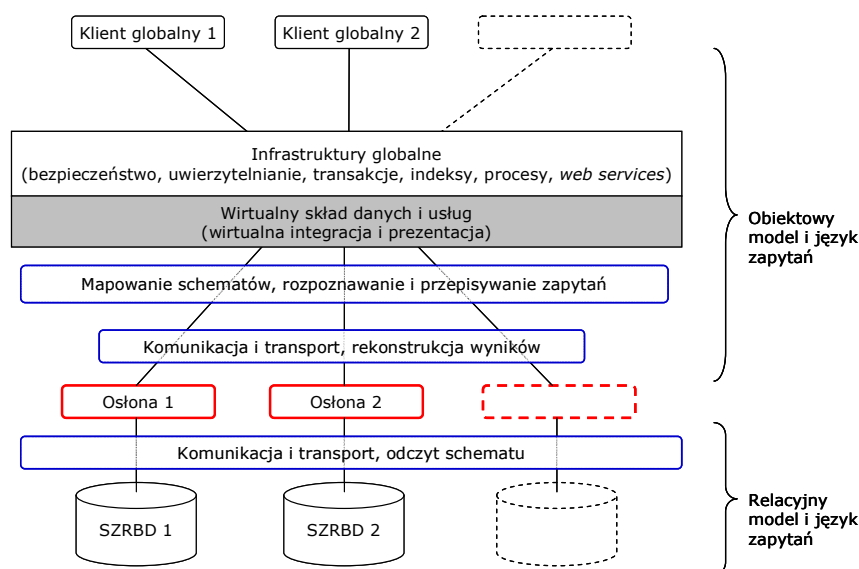
W wirtualnym repozytorium, którego część stanowi opisywana osłona, proces mediacji oparty został o aktualizowane perspektywy obiektowe [14] oparte na koncepcji podejścia stosowego (SBA) [15] (rozdział 4).

## Koncepcja integracji obiektowo-relacyjnej

W kolejnych sekcjach omówione zostały architektura i działanie wirtualnego repozytorium (w którego najniższych warstwach działa osłona) oraz opracowane i zaimplementowane metody integracji i optymalizacji. Na samym końcu umieszczony został przykład takiego procesu zrealizowany w oparciu o prosty schemat relacyjny.

### Wirtualne repozytorium

Rys. 1 przedstawia ogólną architekturę wirtualnego repozytorium z jego podstawowymi elementami funkcjonalnymi. W ogólnym przypadku zasobami integrowanymi przez repozytorium mogą być dowolne dane i usługi, jednak dla uproszczenia przedstawione zostały jedynie bazy relacyjne.

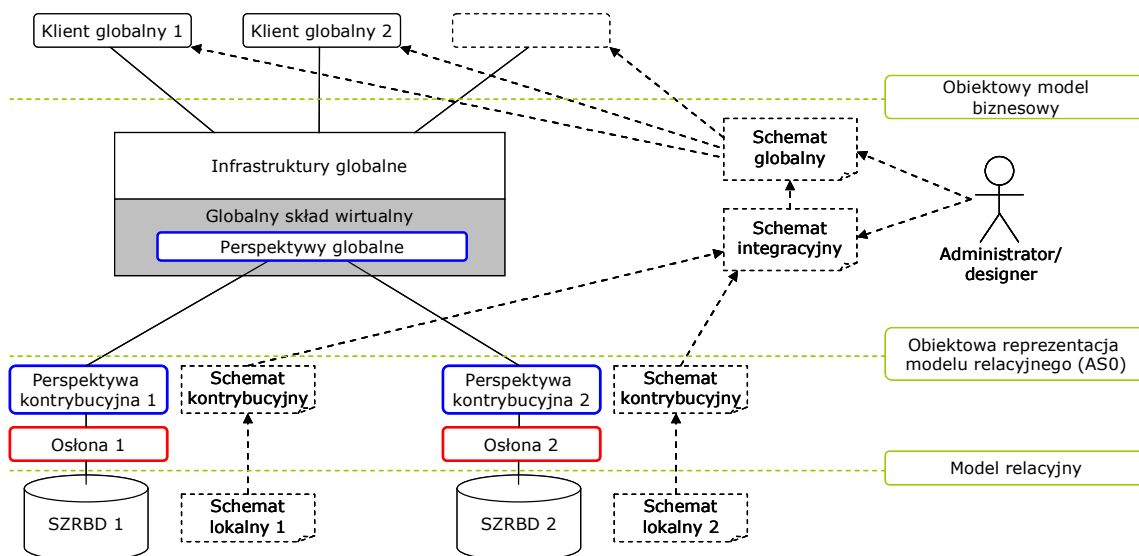


Rys. 1 Ogólna architektura wirtualnego repozytorium

Wirtualne repozytorium udostępnia globalnym klientom wymagane funkcjonalności, np. uwierzytelnianie i bezpieczeństwo oraz mechanizmy komunikacyjne. Jest ono także odpowiedzialne za integrację i zarządzanie danymi wirtualnymi oraz prezentację tych danych zgodnie ze schematem globalnym. Zbiór osłon (zaznaczonych na czerwono) pośredniczy między repozytorium i zasobami. Podstawowe funkcjonalności osłon (zaznaczone na niebiesko) odnoszą się do:

- zapewnienia komunikacji i transportu (zarówno pomiędzy osłoną i wirtualnym repozytorium, jak i pomiędzy osłoną i zasobem),

- umożliwienia (w najlepszym przypadku – zautomatyzowanego) odczytu schematu relacyjnego
- mapowania schematu relacyjnego na model obiektowy,
- analizy i przetwarzania obiektowych zapytań tak aby odpowiednie wyniki zostały zwrócone z zasobów relacyjnych.



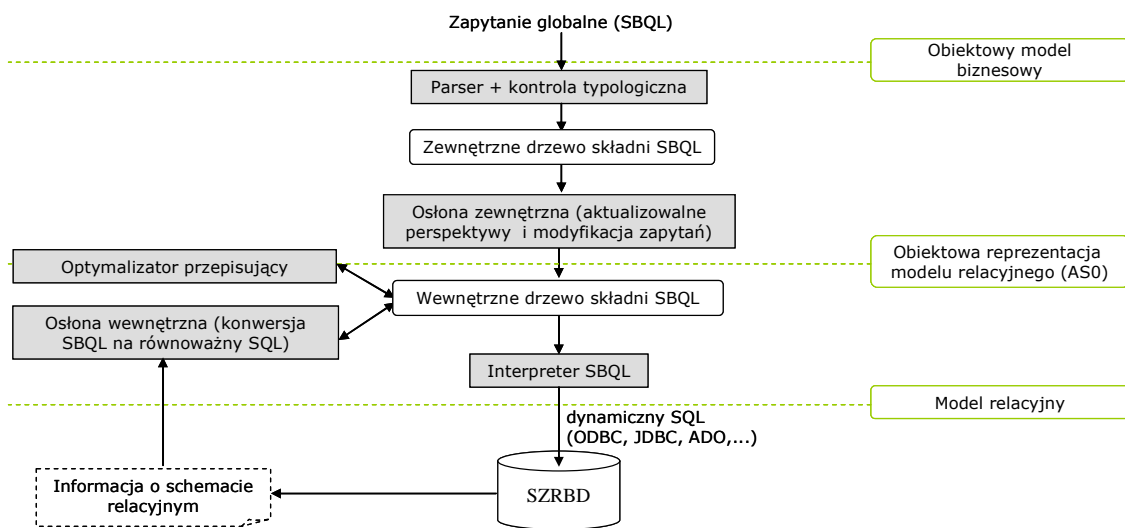
Rys. 2 Integracja schematów w wirtualnym repozytorium

Umieszczony powyżej Rys. 2 przedstawia nieco inny widok repozytorium – wskazane zostały poszczególne warstwy modelu danych pojawiające się podczas integracji i mapowania schematów. Perspektywy kontrybucyjne przedstawiają schematy relacyjne jako proste modele AS0 zdefiniowane w SBA. Stanowią one część schematu globalnego zdefiniowanego i zarządzanego przez administratora (projektanta) repozytorium, co oznacza że muszą one być zgodne z nazwami i strukturami zdefiniowanymi w schemacie integracyjnym. Sam schemat integracyjny jest odpowiedzialny za łączenie schematów lokalnych (zgodnie ze znanymi regułami fragmentacji i ontologiami) w schemat globalny udostępniany użytkownikom globalnym, który jest jedynym schematem widocznym i dostępnym na zewnątrz repozytorium. W najprostszym przypadku schemat integracyjny może zostać wyeksponowany jako schemat globalny, z założenia jednak jest on dalej modyfikowany, aby zgadzał się z wymogami repozytorium. W najbardziej ogólnym przypadku może pojawić się dowolna liczba schematów globalnych, które spełniają wymagania klientów i udostępniają tylko te dane, do których mają uprawnienia. Wszystkie schematy, a w szczególności mapowania i transformacje zachodzące pomiędzy nimi, wyrażone są za pomocą aktualizowalnych perspektyw obiektowych

(schematom integracyjnym i globalnym odpowiadają globalne perspektywy wirtualnego repozytorium).

## Przetwarzanie zapytań i metody optymalizacyjne<sup>1</sup>

Poza mapowaniem i translacją schematów (tak aby zasoby relacyjne stały się dostępne w wirtualnym repozytorium), osłona jest odpowiedzialna za przetwarzanie zapytań. Schemat tego procesu jest przedstawiony na Rys. 3, gdzie modele schematów zaznaczone kolorem jasnozielonym odpowiadają poziomom na Rys. 2 powyżej.



Rys. 3 Integracja schematów w wirtualnym repozytorium

Globalne zapytanie *ad hoc* (odnoszące się do schematu globalnego) pochodzące od jednego z globalnych klientów jest w standardowy sposób parsowane i poddawane kontroli typologicznej, czego efektem jest zewnętrzne drzewo składni (wciąż odnoszące się do schematu globalnego). Na tym drzewie wykonywane jest makropodstawienie definicji perspektyw (odpowiadających schematom globalnym, integracyjnym i kontrybucyjnym przedstawionym na Rys. 2) oraz stosowane są procedury modyfikacji zapytań<sup>2</sup>. Na tym etapie pojawiają się olbrzymie wewnętrzne drzewo składni SBQL odnoszące się do obiektów podstawowych, tj. tych, które są bezpośrednio eksponowane przez osłonę. Następnie stosowane są optymalizatory przepisujące SBQL (rozdział 4.4 pracy) współdziałające z optymalizatorem wewnętrznej osłony. Przepisywacz osłony analizuje drzewo składni w celu znalezienia wyrażeń odpowiadających nazwom

<sup>1</sup> Szczegóły opisane zostały w rozdziale 6

<sup>2</sup> Perspektywy muszą spełniać kilka prostych warunków, aby modyfikacja zapytań mogła być zastosowana (rozdział 4.4.2 pracy), co musi zostać zapewnione przez administratora systemu

„relacyjnym” (dotyczącym relacyjnych tabel i kolumn). Procedura analizy oparta jest na informacji o schemacie relacyjnym, metabazie oraz sygnaturach wyrażeń nadanych podczas kontroli typologicznej.

Jeżeli nazwy „relacyjne” zostają znalezione, zawierające je podzapytania SBQL są zamieniane na odpowiednie wyrażenia dynamicznego SQL (*execute immediately*), które będą ewaluowane w osłanianym zasobie. Zgodnie z podejściem naiwnym, każda nazwa podpowiadająca relacyjnej tabeli powinna zostać zastąpiona przez proste zapytanie SQL *select \* from tabela*. W ten sposób pobrane zostają wszystkie rekordy, natomiast właściwa ewaluacja zapytania wykonywana jest przez wirtualne repozytorium. To podejście jest zawsze *poprawne* i *wiarygodne*, jednak *skrajnie niewydajne*, ponieważ wprowadza niepożądane transport i materializację danych. W związku z tym pojawia się *silna potrzeba optymalizacji*, tak aby główny ciężar ewaluacji zapytań przeniesiony został w dół na osłaniane bazy relacyjne, gdzie mają możliwość działania bardzo skuteczne optymalizatory relacyjne (rozdział 3.3).

### **Podejście naiwne a optymalizacja**

Jak zostało wspomniane, podejście naiwne może zostać zastosowane zawsze, jednak nie pozwala na działanie optymalizatorów relacyjnych. Wyrażenia SBQL pojawiające się w wyniku transformacji (optymalizacji) drzewa składni powinny być dalej analizowane przez osłonę wewnętrzną, aby mogły zostać znalezione jak największe podzapytania (wzorce) transformowalne do optymalizowalnego SQL<sup>3</sup>. Cele optymalizacji osłony są następujące:

- redukcja ilości pobieranych i materializowanych danych (w najbardziej korzystnym przypadku istnieje możliwość pobrania jedynie pożądanych rezultatów),
- minimalizacja przetwarzania po stronie wirtualnego repozytorium (w najbardziej korzystnym przypadku wyniki zgadzają się dokładnie z intencją oryginalnego zapytania).

Optymalizacja osłony jest wyzwaniem znacznie poważniejszym niż proste przepisywanie zastosowane w podejściu naiwnym. Przede wszystkim wiele operatorów

---

<sup>3</sup> Pomimo że optymalizatory SQL (opisane w rozdziale 3.3) są przezroczyste, można z dużym prawdopodobieństwem założyć kiedy zadziałają – na przykład podczas ewaluacji złączeń, selekcji po kolumnach indeksowanych, itp.

i wyrażeń SBQL nie posiada odpowiedników relacyjnych (np. *groupas*), w przypadku wielu innych semantyka jest różna w obydwu językach zapytań (np. operator przypisania w SBQL nie jest makroskopowy). W związku z powyższym podstawową czynnością jest wyizolowanie zbioru operatorów SBQL transformalnych do SQL. Zgodnie z założeniami, optymalizator osłony powinien starać się znaleźć możliwie największe podzapytanie, które może być wyrażone w równoważnym SQL. Dlatego założona została następująca kolejność przeszukiwania:

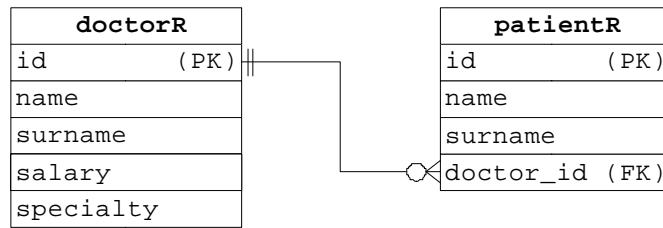
- funkcje agregacyjne,
- złączenia,
- selekcje,
- nazwy odpowiadające tabelom relacyjnym.

Taka kolejność wynika z możliwych form złożonych wyrażeń i ich argumentów (wyrażeń wewnętrznych). Przykładowo, funkcja agregacyjna może być ewaluowana na złączeniach lub selekcjach, same złączenia mogą zawierać warunki selekcji, itp. W przypadku funkcji agregacyjnych, selekcji i nazw tabel istnieje dodatkowa szansa na optymalizację, ponieważ mogą zostać ustalone projekcje i pobrane jedynie wymagane kolumny relacyjne.

Szczegółowe procedury i algorytmy optymalizacyjne osłony (wraz z przykładami rzeczywistych zapytań i dokonywanych przekształceń) omówione zostały w rozdziale 6 pracy. Natomiast dyskusja wyników optymalizacji na zbiorze ponad 40 reprezentatywnych zapytań zawarta jest w rozdziale 7.

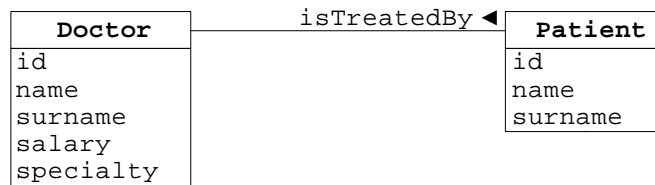
### **Przykład koncepcyjny**

Prezentowany abstrakcyjny (niezależny od implementacji) przykład opiera się na prostym schemacie relacyjnym (Rys. 4). Przedstawiona baza medyczna zawiera dane pacjentów (tabela *patientR*) oraz lekarzy (*doctorR*) – „R” zostało dodane do nazw dla podkreślenia ich „relacyjnego” pochodzenia i zwiększenia czytelności przykładu. Każdy pacjent jest leczony przez jakiegoś lekarza, która to zależność jest odzwierciedlona przez związek klucza podstawowego i obcego na kolumnach *doctorR.id* i *patientR.doctor\_id*. Poza indeksami wynikającymi z istnienia kluczy podstawowych, w schemacie zostały zdefiniowane nieunikatowe wtórne indeksy na kolumnach *patientR.surname* i *doctorR.surname*.



Rys. 4 Schemat relacyjny dla przykładu koncepcyjnego

Ten schemat relacyjny jest importowany przez osłonę i w oparciu o niego tworzone są podstawowe obiekty odpowiadające relacyjnym tabelom i kolumnom – tabela jest odzwierciedlona jako obiekt złożony z podobiektami odpowiadającymi kolumnom oraz ich prymitywnym typom danych. Zastosowano tu mapowanie jeden-do-jednego, włącznie z zachowaniem nazw relacyjnych względem tworzonego schematu obiektowego. Powstający w ten sposób prosty schemat obiektowy jest już gotowy do odpytywania, jednak wciąż nie odzwierciedla relacyjnych związków i więzów integralności, dlatego w kolejnym kroku zostaje pokryty aktualizowanymi perspektywami obiektowymi. Końcowa forma została przedstawiona na Rys. 5 – związek kluczy podstawowych i obcych jest odzwierciedlony przez wirtualny wskaźnik *isTreatedBy*.



Rys. 5 Schemat obiektowy dla przykładu koncepcyjnego

Poniżej znajduje się zredukowany kod perspektyw użytych przy definicji schematu. Dla uproszczenia nie zostały zdefiniowane procedury aktualizacji danych, jedynie ich pobieranie:

```

view DoctorDef {
    virtual objects Doctor: record {d: doctorR;}[0..*] {
        return (doctorR) as d;
    }
    /* on_retrieve pominięte dla Doctor */
}
view idDef {
    virtual objects id: record {_id: doctorR.id;} {
        return d.id as _id;
    }
    on_retrieve: integer {
        return deref(_id);
    }
}
view nameDef {
    virtual objects name: record {_name: doctorR.name;} {

```

```

        return d.name as _name;
    }
    on_retrieve: string {
        return deref(_name);
    }
}
view surnameDef {
    virtual objects surname: record {_surname: doctorR.surname;} {
        return d.surname as _surname;
    }
    on_retrieve: string {
        return deref(_surname);
    }
}
view salaryDef {
    virtual objects salary: record {_salary: doctorR.salary;} {
        return d.salary as _salary;
    }
    on_retrieve: real {
        return deref(_salary);
    }
}
view specialtyDef {
    virtual objects specialty: record {_specialty: doctorR.specialty;} {
        return d.specialty as _specialty;
    }
    on_retrieve: string {
        return deref(_specialty);
    }
}
}

view PatientDef {
    virtual objects Patient: record {p: patientR;}[0..*] {
        return (patientR) as p;
    }
    /* on_retrieve pominięte dla Patient */
}
view idDef {
    virtual objects id: record {_id: patientR.id;} {
        return p.id as _id;
    }
    on_retrieve: integer {
        return deref(_id);
    }
}
view nameDef {
    virtual objects name: record {_name: patientR.name;} {
        return p.name as _name;
    }
    on_retrieve: string {
        return deref(_name);
    }
}
view surnameDef {
    virtual objects surname: record {_surname: patientR.surname;} {
        return p.surname as _surname;
    }
    on_retrieve: string {
        return deref(_surname);
    }
}
view isTreatedByDef {
    virtual objects isTreatedBy: record {_isTreatedBy: patientR.doctor_id;} {
        return p.doctor_id as _isTreatedBy;
    }
    on_retrieve: integer {
        return deref(_isTreatedBy);
    }
}

```

```

    }
    on_navigate: Doctor {
      return Doctor where id = _isTreatedBy;
    }
  }
}

```

Przykładowe zapytanie, które może zostać zadane dla takiego schematu to “zwróć nazwiska lekarzy leczących pacjentów o nazwisku Smith, których pensja jest równa minimalnej pensji kardiologa”. Postać zapytania w SBQL:

```

((Patient where surname = "Smith").isTreatedBy.Doctor as doc where doc.salary
= min((Doctor where specjalty = "cardiology").salary)).doc.surname;

```

Pierwszym krokiem przekształceń jest wprowadzenie jawnych dereferencji w miejscach, gdzie jest to konieczne:

```

(((((((Patient where (deref(surname) = "Smith")) . isTreatedBy) . Doctor)) as
doc where (deref((doc . salary)) = min(deref(((Doctor where (deref(specialty)
= "cardiology")) . salary)))) . doc) . surname);

```

Następnie, wywołania *deref* zamieniane są (makropodstawienie) na odpowiednie definicje *on\_retrieve* i *on\_navigate* odpowiednio dla wirtualnych obiektów i wirtualnych wskaźników. Wywołania perspektyw są podstawiane zapytaniami z definicji worków (*sacks*). Ten krok pozwala na zastosowanie modyfikacji zapytań, ponieważ definicje perspektyw składają się z pojedynczego zapytania:

```

(((((((patientR as p where (((p . surname)) as _surname . deref(_surname))
= "Smith")) . ((p . doctor_id)) as _isTreatedBy) . ((doctorR as d where (((d
. id)) as _id . deref(_id)) = deref(_isTreatedBy)))) as doc where (((doc .
((d . salary)) as _salary) . deref(_salary)) = min((((doctorR as d where
((((d . specjalty)) as _specialty . deref(_specialty)) = "cardiology")) . ((d
. salary)) as _salary) . deref(_salary)))) . doc) . ((d . surname)) as
_surname);

```

Teraz usuwane zostają (gdzie jest to możliwe) nazwy pomocnicze (*p*, *d*, *\_surname*, *\_isTreatedBy*, *\_id*, *\_salary*, *\_specialty*, *\_salary*) – dzięki modyfikacji zapytań definicje makropodstawione w poprzednim kroku stanowią regularną część zapytania i przekształcenia składniowe są poprawne:

```

(((((((doctorR where (deref(id) = ((patientR where (deref(surname) = "Smith"))
. deref(doctor_id)))) as doc where ((doc . deref(salary)) = min(((doctorR
where (deref(specialty) = "cardiology")) . deref(salary)))) . doc) .
surname)) as _surname;

```

W kolejnym kroku stosowana jest optymalizacja SBQL – w prezentowanym przykładzie mogą zostać znalezione dwa niezależne podzapytania (dla znalezienia minimalnej pensji kardiologa i pacjentów o nazwisku Smith). Podzapytania te zostają „wyciągnięte” przed zapytanie i zastąpione pomocniczymi nazwami *aux0* i *aux1*:

```

((((((min(((doctorR where (deref(specialty) = "cardiology")) . deref(salary)))
as aux0 . ((((((patientR where (deref(surname) = "Smith")) .
deref(doctor_id))) as aux1 . (doctorR where (deref(id) = aux1)))) as doc where
((doc . deref(salary)) = aux0))) . doc) . surname)) as _surname;

```

Na tej postaci zapytania wykonywane są przez osłonę analiza i optymalizacja. W oparciu o dostępne informacje o modelu relacyjnym mogą zostać utworzone następujące zapytania SQL wywoływane przez *execute immediately*:

```
exec_immediately("select min(salary) from doctorR where specjalty =  
'cardiology'") as aux0 . exec_immediately("select doctor_id from patientR  
where surname = 'Smith'") as aux1 . exec_immediately("select surname from  
doctorR where salary = '" + aux0 + "' and id = '" + aux1 + "'") as _surname;
```

Każde z nich zostanie wykonane przez bazę relacyjną z wykorzystaniem odpowiednich indeksów. Przetwarzanie w wirtualnym repozytorium ograniczone zostanie do zmagazynowania cząstkowych wyników z dwóch pierwszych zapytań SQL, które zostaną wrzucone na stos w celu sparametryzowania ostatniego zapytania SQL zwracającego wynik dokładnie realizujący pierwotne zapytanie.

## Podsumowanie

Opracowana i zaimplementowana procedura importu schematów relacyjnych pozwala na generyczną, zautomatyzowaną i całkowicie przeźroczystą integrację dowolnej liczby spadkowych baz relacyjnych do struktur wirtualnego repozytorium. Schematy relacyjne są przedstawiane, udostępniane i przetwarzane nieodróżnialnie od rzeczywistych danych obiektowych. Importowany schemat jest przykrywany aktualizowanymi obiektowymi perspektywami zdefiniowanymi w języku SBQL, który wykorzystywany jest do zarządzania i konserwacji wirtualnego repozytorium, a także jako interfejs dla klientów globalnych. Dzięki temu osłaniane bazy relacyjne są przetwarzane przeźroczyście jak dowolne inne zasoby wirtualnego repozytorium. Faktyczne odróżnienie od prawdziwych danych obiektowych ma miejsce dopiero na poziomie osłony, poniżej perspektyw kontrybucyjnych – żaden wyższy element wirtualnego repozytorium nie jest „świadom” rzeczywistego charakteru odpytywanego zasobu.

Osłona, pośrednicząc pomiędzy wirtualnym repozytorium a bazą relacyjną, przeprowadza analizę zapytań SQL pod kątem znalezienia nazw relacyjnych i zawierających je podzapytań. Wyszukane zostają możliwie największe podzapytania, które są zastępowane odpowiednimi wyrażeniami SBQL przekierowującymi napisy zapytań SQL do odpowiednich osłon i baz relacyjnych. Wyniki otrzymywane z zasobu są przekształcane do postaci obiektów i przekazywane na stopy, dzięki czemu mogą być dalej przetwarzane jak inne dane (lub zwrócone bezpośrednio do użytkownika). Optymalizator przepisujący osłony przeprowadza także transformacje pozwalające na wykonywanie imperatywnych konstrukcji SBQL – aktualizację danych relacyjnych zgodnie z semantyką obiektowego języka zapytań.

## Bibliografia<sup>4</sup>

- 1 Kuliberda K., Wiślicki J., Adamus R., Subieta K.: *Object-Oriented Wrapper for Relational Databases in the Data Grid Architecture, On the Move to Meaningful Internet Systems 2005 Proceedings. LNCS 3762, Springer 2005, pp. 367-376*
- 2 Wiślicki J., Kuliberda K., Adamus R., Subieta K.: *Relational to Object-Oriented Database Wrapper Solution in the Data Grid Architecture with Query Optimization Issues, IBM Research Report RC23820 (W0512-007), Proceedings SOBPI'05 (ICSOC'05), Amsterdam, Holland, 2005, pp. 30-43*
- 3 Adamus R., Kuliberda K., Wiślicki J., Subieta K.: *Wrapping Relational Data Model to Object-Oriented Database in the Data Grid Architecture, SOFSEM SRF 2006 Proceedings, Merin, Czech Republic, 2006, pp. 54-63*
- 4 Wiślicki J., Kuliberda K., Kowalski T., Adamus R.: *Integration of Relational Resources in an Object-Oriented Data Grid, SiS 2006 Proceedings, Łódź, Poland, 2006, pp. 277-280*
- 5 Wiślicki J., Kuliberda K., Kowalski T., Adamus R.: *Integration of relational resources in an object-oriented data grid with an example, Journal of Applied Computer Science (2006), Vol. 14 No. 2, Łódź, Poland, 2006, pp. 91-108*
- 6 Wiślicki J., Kuliberda K., Adamus R., Subieta K.: *Relational to object-oriented database wrapper solution in the data grid architecture with query optimization issues, IJBPM, 2007/2008 (to appear)*
- 7 Codd E. F.: *A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387*
- 8 Atkinson M. et al.: *The Object-Oriented Database System Manifesto, Proc. of 1st Intl. Conf. on Deductive and OO Databases 89, Kyoto, Japan, 1989, pp. 40-57*
- 9 Wiederhold G.: *Mediators in the Architecture of Future Information Systems, IEEE Computer, 25(3), 1992, pp. 38-49*
- 10 Ahmed R. et al.: *An overview of Pegasus, In: Proceedings of the Workshop on Interoperability in Multidatabase Systems, RIDE-IMS'93, Vienna, Austria, 1993*
- 11 Fahl G., Risch T.: *Query processing over object views of relational data, The VLDB Journal (1997) 6: 261-281*
- 12 Risch T. et al.: *Functional data integration in a distributed mediator system, In Functional Approach to Computing with Data, P.Gray, L.Kerschberg, P.King, and A.Poulovassilis, Eds. Springer, 2003*
- 13 Tomasic A. et al.: *The Distributed Information Search Component Disco and the World Wide Web, SIGMOD Conference 1997, pp. 546-548, 1997*
- 14 Kozankiewicz H., Leszczyłowski J., Subieta K.: *Implementing Mediators through Virtual Updateable Views, Engineering Federated Information Systems, Proceedings of the 5th Workshop EFIS 2003, July 17-18 2003, Coventry, UK, pp.52-62*
- 15 Subieta K.: *Theory and Construction of Object-Oriented Query Languages. PJIIT - Publishing House, ISBN 83-89244-28-4, 2004, 522 pages (in Polish)*

---

<sup>4</sup> Kompletna bibliografia pracy doktorskiej zawiera 223 pozycje