



Polsko-Japońska Wyższa Szkoła Technik Komputerowych

JPS

Języki i Środowiska Programowania Baz Danych 2009/2010

PJWSTK, 2009

Marcin Dąbrowski

mdabrowski@pjwstk.edu.pl



Wprowadzenie

Wykład i ćwiczenia z JPS są poświęcone obiektowym bazom danych. Studenci poznają podstawowe założenia podejścia stosowego (SBA, Stack-Based Approach) wykorzystywanego w obiektowych bazach danych. Prezentowany zostanie język SBQL (Stack Based Query Language), wykorzystywany w systemie ODRA, zarówno od strony użytkowej, semantyki jak i implementacyjnej. Zdobyta wiedza powinna posłużyć do zbudowania własnego mechanizmu obiektowej bazy danych z własnym językiem zapytań opartym o zasady SBA.

Ćwiczenia

Ćwiczenia z JPS w roku 2009/2010 w semestrze zimowym odbywać się będą:

Dla grupy G – czwartek, godzina 15:30, sala 132

Dla grupy A2 – czwartek, godzina 17:00, sala 132

Dla grupy A1 – wtorek, godzina 15:30, sala 114

Dla grupy OI1 – wtorek, godzina 17:00, sala 114

Dla grupy IO2 – wtorek, godzina 19:00, sala 114

Zaliczenie ćwiczeń i system ocen

Podczas semestru, każdy ze studentów zobowiązany jest przedstawiać w kolejnych terminach mini-projekty które ostatecznie złożą się na duży projekt. Składowe oceny są następujące:

Cały projekt	20pkt
Mini-projekt 1 (DataStore)	12pkt
Mini-projekt 2 (QueryResult)	12pkt
Mini-projekt 3 (ENVS)	12pkt
Mini-projekt 4 (Visitor)	8pkt
Mini-projekt 5 (AST)	12pkt
Mini-projekt 6 (Operator)	12pkt
Mini-projekt 7 (Lekser i parser)	12pkt



Obowiązkowe jest oddanie wszystkich mini-projektów **z wyjątkiem ostatniego**. Osoby które oddadzą projekt składający się z 6 mini-projektów mogą liczyć najwyżej na ocenę 4.5 na koniec ćwiczeń. Przewiduje się standardową drabinkę ocen tzn.

5,0 – > 90pkt

4,5 – 81-90pkt

4,0 – 71-80pkt

3,5 – 61-70pkt

3,0 – 51-60pkt

2,0 – <51pkt

Terminarz oddawania projektów

Każdy mini-projekt należy oddawać w terminie, zgodnie z kalendarzem poniżej. **Każdy tydzień zwłoki oznacza trzy punkty mniej możliwe do uzyskania.**

I. tydzień	Ćwiczenia
II. tydzień	Ćwiczenia
III. tydzień	Ćwiczenia
IV. tydzień	Oddanie projektu-mini 1 (DataStore)
V. tydzień	Ćwiczenia
VI. tydzień	Oddanie projektu-mini 2 (QResult)
VII. tydzień	Ćwiczenia
VIII. tydzień	Oddanie projektu mini-3 (ENVS)
IX. tydzień	Oddanie projektu mini-4(Visitor)
X. tydzień	Ćwiczenia
XI. tydzień	Oddanie projektu mini-5(AST)
XII. tydzień	Oddanie projektu mini-6(Operator)
XIII. tydzień	Oddanie projektu mini-7(Lekser i parser)
XIV. tydzień	Oddanie projektów końcowych
XV. tydzień	Wpisy

ZAŁOŻENIA PROJEKTU

W ramach ćwiczeń każdy student zobowiązany jest przedstawić prototyp systemu obiektowej bazy danych opartej o SBA. Baza danych powinna być w stanie wczytać do pamięci dane przechowywane w pliku XML a następnie odpowiadać na zapytania do niej kierowane zawierające kilka podstawowych operatorów. Całe rozwiązanie może być prostą aplikacją konsolową. W przypadku gdy student nie zdecyduje się na stworzenie parsera i leksera konieczne jest stworzenie mechanizmu pozwalającego przekazywanie do programu drzewa AST dowolnego zapytania.



Przydział operatorów studentom w ramach grup odbędzie się podczas jednych z kolejnych ćwiczeń.

Projekt może być zrealizowany w języku Java lub C#. Każdy inny język programowania musi zostać uzgodniony z prowadzącym zajęcia. W szczególności należy się upewnić co do dostępności dla niego narzędzi potrzebnych do wykonania leksera i parsera – flex i bison.

MINI-PROJEKTY

1. DataStore

Należy utworzyć strukturę klas zdolną przechowywać w pamięci programu dane, zgodnie z postulatami SBA. Klasy muszą być wyposażone w zestaw metod pozwalających na tworzenie, aktualizowanie i usuwanie utrzymywanych danych. Ponadto należy stworzyć mechanizm pozwalający na odczyt danych wejściowych z pliku XML. Można się w tym celu posłużyć się narzędziami dostępnymi w danej platformie (JAXB dla Javy czy API System.XML dla .NET) Zakładamy, że dane w pliku są zgodne z modelem M0. Przykładowy plik:

```
<baza>
  <test>
    10
  </test>
  <osoba>
    <numer>s0000</numer>
    <grupa>123</grupa>
    <spec>IO</spec>
    <imie>Anna</imie>
    <nazwisko>Kowalska</nazwisko>
    <adres>
      <ulica>Jerozolimskie</ulica>
      <kod>00-000</kod>
      <miasto>Warszawa</miasto>
    </adres>
    <ksiazka>
      <autor>Adam Mickiewicz</autor>
      <tytul>Pan Tadeusz</tytul>
      <egzemplarz>1234</egzemplarz>
      <bibliotekarz>Hanna Kowalska</bibliotekarz>
    </ksiazka>
  </osoba>
  <osoba>
    <numer>s0001</numer>
    <grupa>123</grupa>
    <spec>IO</spec>
    <imie>Jan</imie>
    <nazwisko>Nowak</nazwisko>
    <adres>
      <ulica>Koszykowa 15</ulica>
      <kod>03-121</kod>
      <miasto>Warszawa</miasto>
    </adres>
    <ksiazka>
      <autor>Juliusz Slowacki</autor>
      <tytul>Kordian</tytul>
      <egzemplarz>1235</egzemplarz>
      <bibliotekarz>Hanna Kowalska</bibliotekarz>
    </ksiazka>
  </osoba>
</baza>
```



2. QueryResult

Należy stworzyć strukturę klas zdolną przechowywać wszystkie przewidziane przez SBA możliwe rezultaty zapytań.

Dodatkowo należy stworzyć klasy odpowiedzialne za stos rezultatów (QRES). Prezentację możliwości stosu należy pokazać poprzez zapisanie w kodzie operacji odpowiednich dla protego wyrażenia algebraicznego (np. dodawanie i odejmowanie od siebie kolejno kilku liczb).

3. ENVS

Należy przygotować klasy pozwalające realizować stos środowiskowy zgodnie z SBA. Dodatkowo należy stworzyć metody realizujące metody: bind i nested. Zasadę działania należy pokazać implementując operator niealgebraiczny kropki i symulując w kodzie zachowanie stosu dla zapytania typu obiekt.podobiekt (np. „osoba.numer”).

4. Visitor

Należy przedstawić zasadę działania wzorca projektowego Visitor (Wizytator). Przykład powinien być możliwie złożony. Na dowolnej strukturze (np. drzewo rezultatów SBQL, przykładowo zbiór w którym znajdują się struktury składające się z różnych elementów prostych typu int, double itp.) powinny operować co najmniej dwaj różni wizytatorzy (dla naszego przykładu jeden drukujący wynik jako prosty tekst a drugi w postaci XML)

5. AST

Należy przygotować klasy pozwalające przechowywać drzewo składniowe zapytania. Poprawność rozwiązania należy udowodnić budując przykładowe drzewo w kodzie.

6. Operatory

Drzewo AST należy rozszerzyć o klasy reprezentujące odpowiednie jedno- i dwuargumentowe operatory. Do obowiązkowych należy kropka (z mini-projektu trzeciego), where oraz podstawowe algebraiczne (dodawanie, odejmowanie, mnożenie, dzielenie). Dodatkowe operatory zostaną przydzielone na zajęciach. Dostępne są:

Algebraiczne:

>, <, =, <> (różne), or, and

przecinek, bag, sequence,

union, minus, intersect, in

sum, avg, min, max, count

as, groupas,

Nie-algebraiczne:

forall, forany, orderby, join

Dodatkowo należy stworzyć, korzystając ze wzorca Visitor, wizytatora który będzie przechodził przez drzewo AST i korzystając z ENVS i QRES dokona ewaluacji zapytania.



Poprawność rozwiązania należy udowodnić wywołując wyliczenie dla drzewa stworzonego w kodzie (podobnie jak w mini-projekcie 5.)

7. Lekser i parser

Należy opracować lekser i parser oraz przerobić projekt tak, by potrafił wczytywać zapytania z konsoli i by generowanie drzewa AST tego zapytania odbywało się automatycznie na podstawie gramatyki parsera.

MATERIAŁY

1. Książka:
K.Subieta. Teoria i konstrukcja obiektowych języków zapytań. Wydawnictwo PJWSTK, Warszawa 2004, ISBN 83-89244-29-2
2. Strona WWW z informacjami o SBA i SBQL
www.sbql.pl
3. Wykłady prof. Subiety w wersji elektronicznej:
www.si.pjwstk.edu.pl/dydaktyka/JPS/
4. Opis i podręcznik użytkownika ODRY
www.ipipan.eu/staff/k.subieta/SBA_SBQL/various/ODRA/ODRA_manual.html

KONTAKT

mdabrowski@pjwstk.edu.pl

me@marcindabrowski.net

(proszę o „JPS” w tytule maila by nie trafił do spamu)