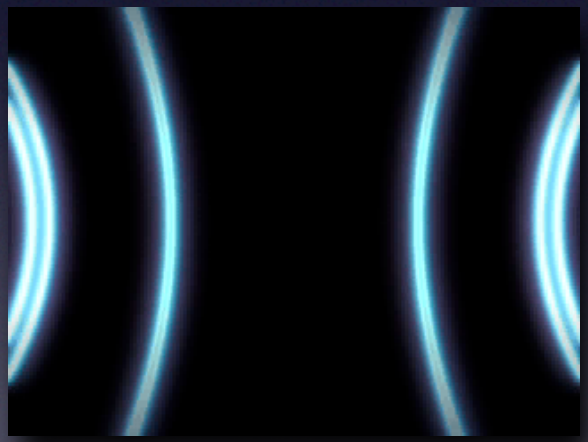
JPS cwiczenia 1.

Wprowadzenie

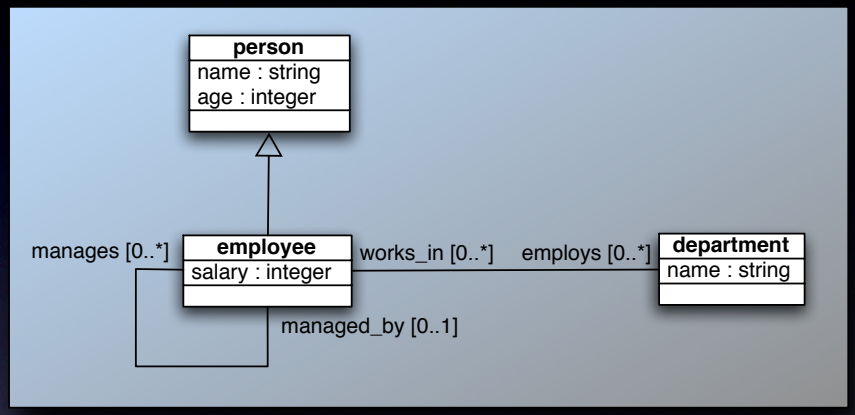


Polsko-Japońska Wyższa Szkoła Technik Komputerowych

Terminologia

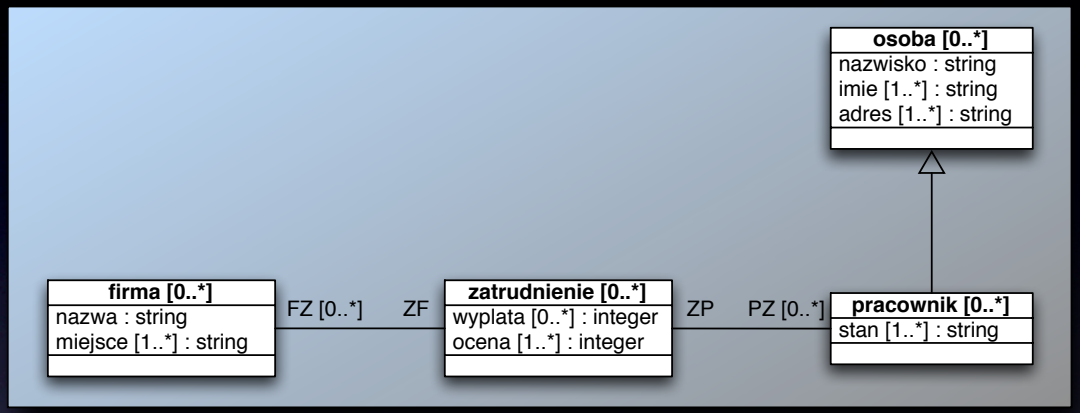
- **SBA**
  Teoretyczna baza dla języków zapytań. Język zapytań jest formą języka programowania.
- **SBQL**
  Język zapytań do sieciowych, hierarchicznych, relacyjnych, obiektowych, XML-owych, etc. baz danych. Po rozszerzeniu o konstrukcje proceduralne powstał pełny język programowania a'la PL/SQL, w którym wyrażenia są zapytaniami.

SBQL jako język zapytań



- nazwiska i wiek pracowników zarabiających więcej niż 10000:
  (employee where salary > 10000).(name, age)
- pracownicy którzy pracują w departamencie Sales:
  employee where “Sales” in works_in.department.name
- szef Smitha:
  (employee where name = “Smith”).managed_by.employee
- departamenty zatrudniające więcej niż 3 pracowników:
  department where count(employs) > 3

SBQL vs SQL

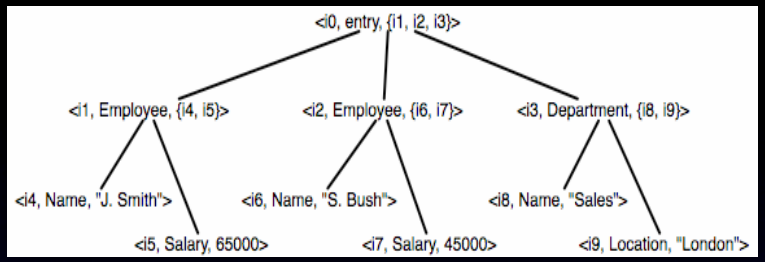


Podaj nazwiska i stanowiska pracowników pracujących w firmach zlokalizowanych w Radomiu:

- SQL:
  select *s.Nazwisko*, *w.Stan*
  from *Firma* **as** *f*, *Lokal* **as** *k*, *Zatrudnienie* **as** *z*,
  Pracownik **as** *p*, *Wyszkolenie* **as** *w, Osoba* **as** s
  where *k.Miejsce* = "Radom" **and** *k.NrF = f.NrF*
  and *f.NrF = z.ZF* **and** *z.ZP = p.NrP* **and** *w.NrP = p.NrP*
  and *p.NrOs = s.NrOs*

- SBQL
  (*Firma* **where** "Radom" **in** *Miejsce*). *FZ.Zatrudnienie.ZP.Pracownik.*(*Nazwisko, Stan*)

Stosy



Employee where Name = "J. Smith" and Salary > 10000

1. Initialize ENVS and QRES.

Employee(i1), Employee(i2), Department(i3)

2. Execute **bind**(Employee)

Employee(i1), Employee(i2), Department(i3)	bag(i1, i2)

2. Pop one element from QRES.

Employee(i1), Employee(i2), Department(i3)

3. Create a new ENVS section. Execute **nested**(i1).

Name(i4), Salary(i5)

Employee(i1), Employee(i2), Department(i3)

4. Execute **bind**(Name)

Name(i4), Salary(i5)	
Employee(i1), Employee(i2), Department(i3)	i4

5. push("J. Smith")

Name(i4), Salary(i5)	"J. Smith"
Employee(i1), Employee(i2), Department(i3)	i4

6. Pop two elements, dereference i4, compare "J. Smith" and "J. Smith", push true.

Name(i4), Salary(i5)	
Employee(i1), Employee(i2), Department(i3)	true

7. Execute bind("Salary")

Name(i4), Salary(i5)	i5

itd.

Projekt

bazka.xml

```
<baza>
        <pracownik>
                <imie>Jan</imie>
                <nazwisko>Kowalski</nazwisko>
                <wiek>53</wiek>
        <pracownik>
        <pracownik>
                <imie>Stefan</imie>
                <nazwisko>Nowak</nazwisko>
                <wiek>33</wiek>
                <adres>
                        <ulica>Szybka</ulica>
                        <numer>5</numer>
                </adres>
        <pracownik>
        <pracownik>
                <imie>Grzegorz</imie>
                <nazwisko>Zdebel</nazwisko>
                <wiek>41</wiek>
                <adres>
                        <ulica>Krotka</ulica>
                        <numer>2</numer>
                </adres>
        <pracownik>
</baza>
```

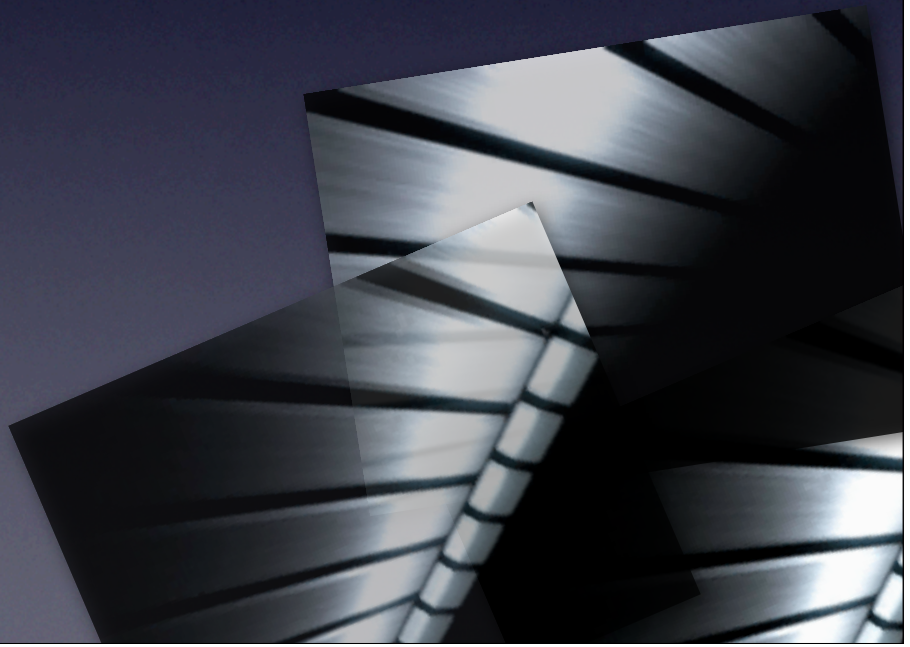
Terminal — bash — 84x27

```
Last login: Thu Mar  3 10:58:09 on ttyp2
Welcome to Darwin!
Ogryzek:~ raist$ cd ~/Desktop/
Ogryzek:~/Desktop raist$ ./sbql bazka.xml
SBQL> (baza.pracownik where nazwisko = "Zdebel").adres;

<adres>
 <ulica>Krotka</ulica>
 <numer>2</numer>
</adres>
Ogryzek:~/Desktop raist$
```

Składniki projektu do zrobienia

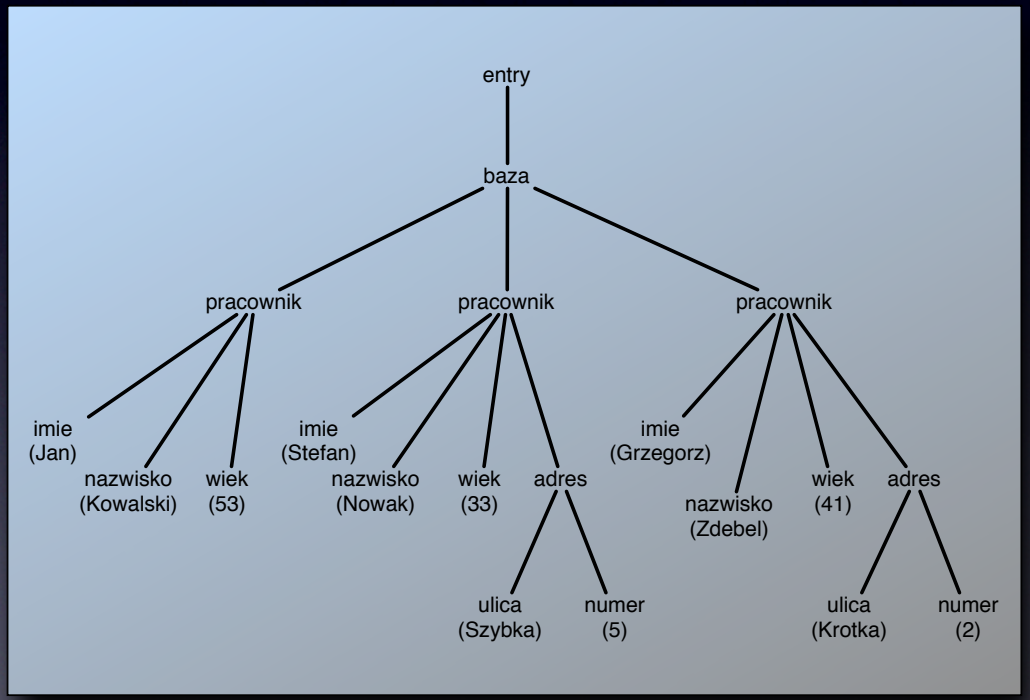
- Skład danych
- Lekser i parser
- Drzewo składniowe (AST) i stosy
- Interpreter



Skład danych

bazka.xml

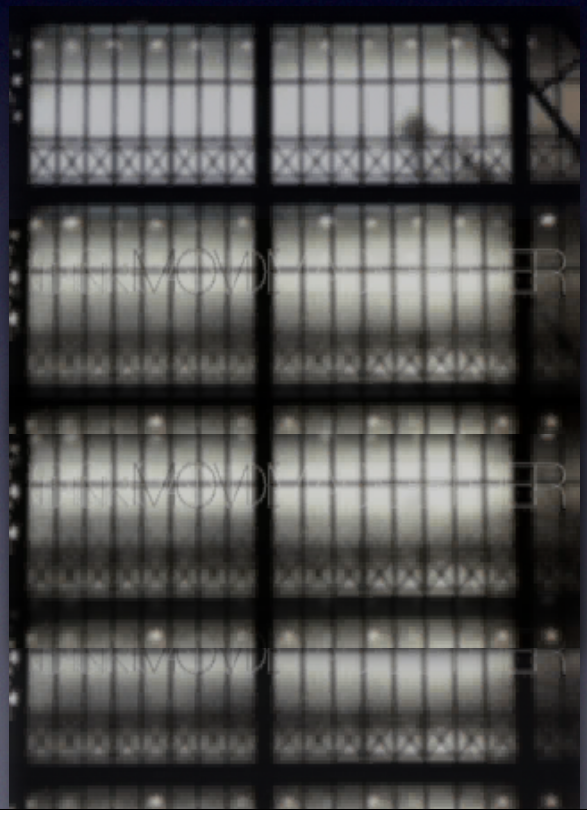
```
<baza>
        <pracownik>
                <imie>Jan</imie>
                <nazwisko>Kowalski</nazwisko>
                <wiek>53</wiek>
        <pracownik>
        <pracownik>
                <imie>Stefan</imie>
                <nazwisko>Nowak</nazwisko>
                <wiek>33</wiek>
                <adres>
                        <ulica>Szybka</ulica>
                        <numer>5</numer>
                </adres>
        <pracownik>
        <pracownik>
                <imie>Grzegorz</imie>
                <nazwisko>Zdebel</nazwisko>
                <wiek>41</wiek>
                <adres>
                        <ulica>Krotka</ulica>
                        <numer>2</numer>
                </adres>
        <pracownik>
</baza>
```



Lekser

baza.pracownik where nazwisko = “Zdebel”

- Identyfikator (”baza”)
- Operator **.**
- Identyfikator (”pracownik”)
- Operator **where**
- Identyfikator (”nazwisko”)
- Operator algebraiczny (**=**)
- Literał łańcucha znaków (”Zdebel”)

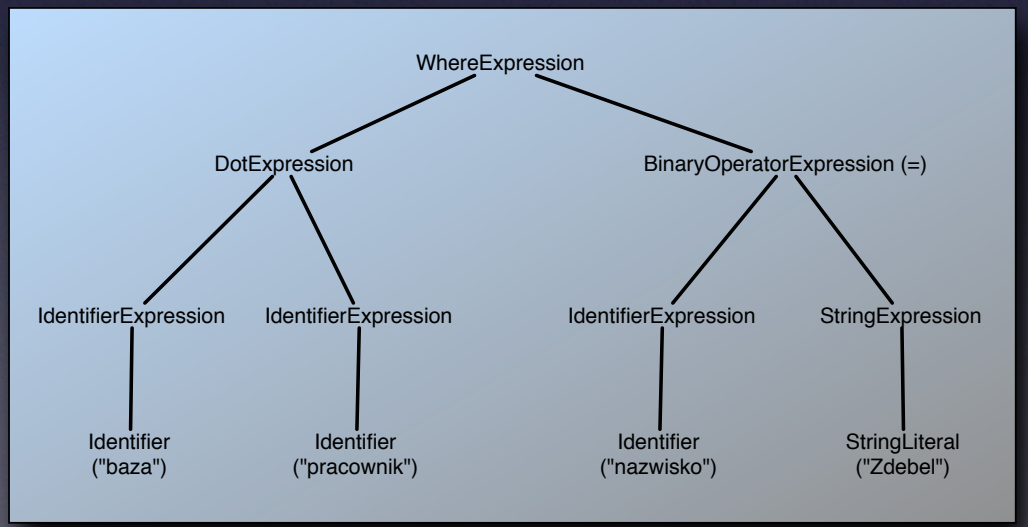


Parser

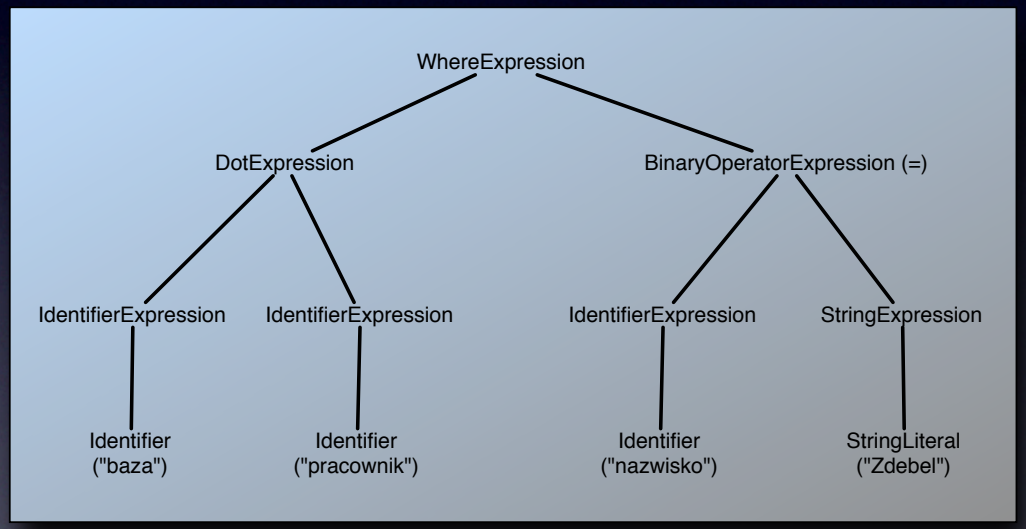
baza.pracownik where nazwisko = “Zdebel”

```
program  ::=
          wyrazenie
        |  instrukcja
        ;

wyrazenie ::= wyrazenie where wyrazenie
        |  wyrazenie . wyrazenie
        |  wyrazenie where wyrazenie
        |  wyrazenie = wyrazenie
        |  wyrazenie + wyrazenie
        |  identyfikator
        |  literal
        |  ( wyrazenie )
      ...
```



Interpreter



Podział na grupy

- grupy max 4 osób
- wybór modułu interpretera
- wybór języka programowania