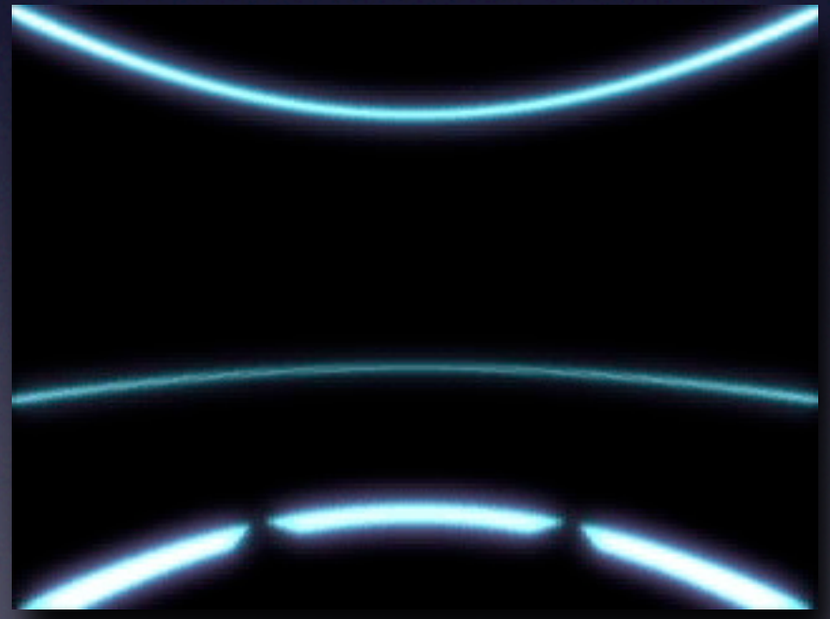
JPS cwiczenia 3.

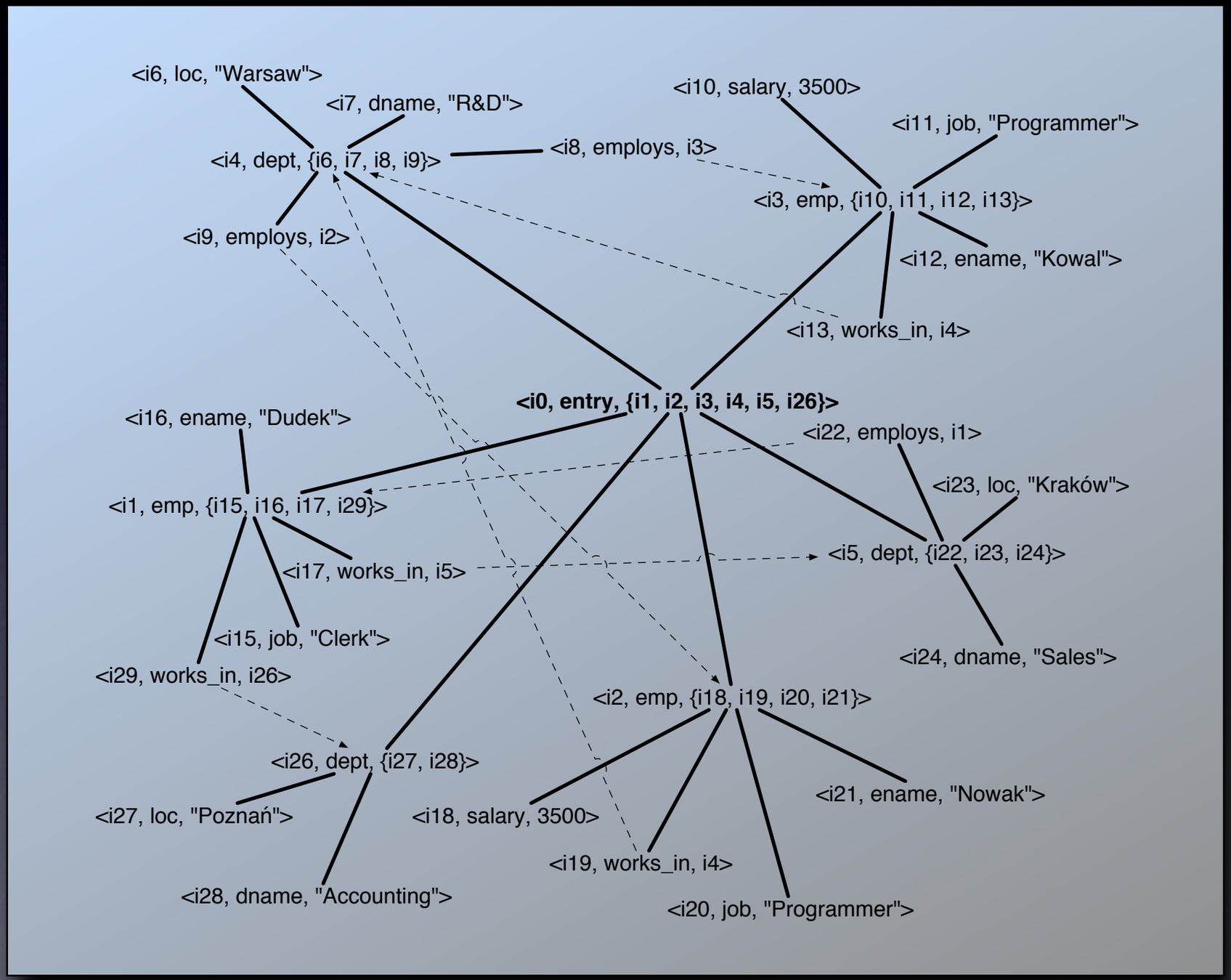
Stosy, proste zapytania



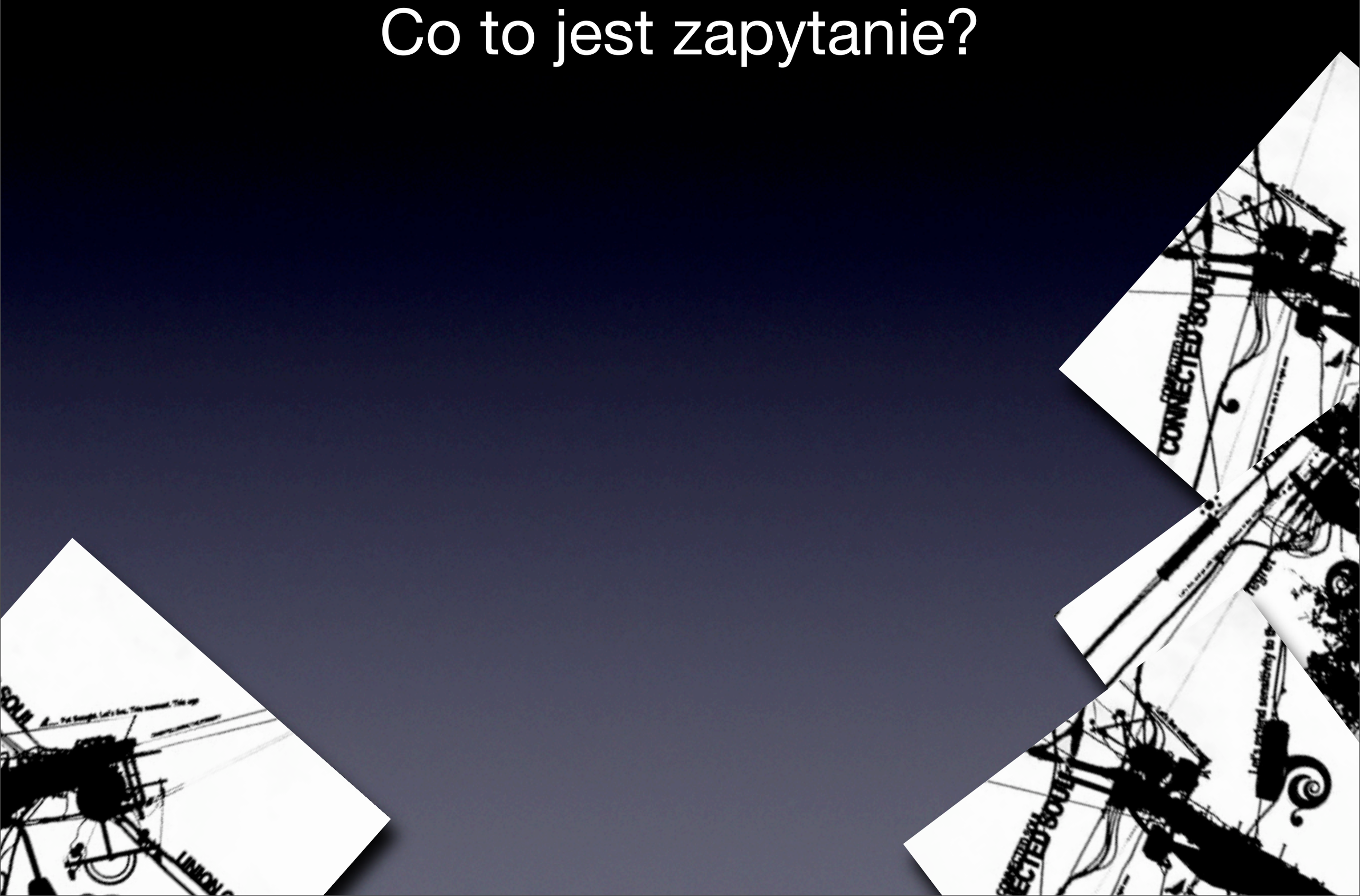
© Michał Lentner
Polsko-Japońska Wyższa Szkoła Technik Komputerowych

Baza danych używana w przykładach

Baza danych używana w przykładach



Co to jest zapytanie?

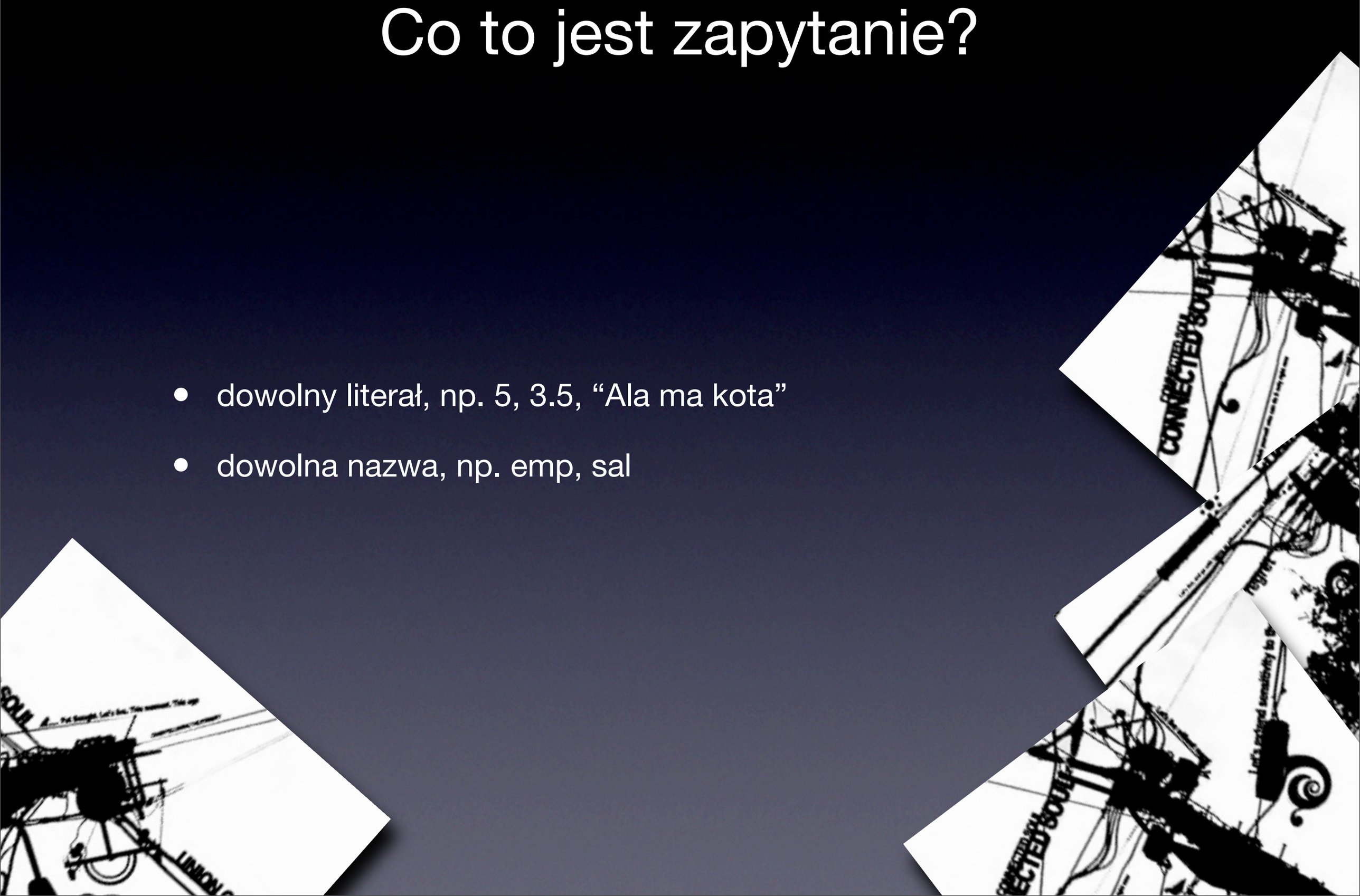


Co to jest zapytanie?

- dowolny literał, np. 5, 3.5, “Ala ma kota”

Co to jest zapytanie?

- dowolny literał, np. 5, 3.5, “Ala ma kota”
- dowolna nazwa, np. emp, sal



Co to jest zapytanie?

- dowolny literał, np. 5, 3.5, “Ala ma kota”
- dowolna nazwa, np. emp, sal
- jedno lub dwa zapytania połączone z jakimś operatorem

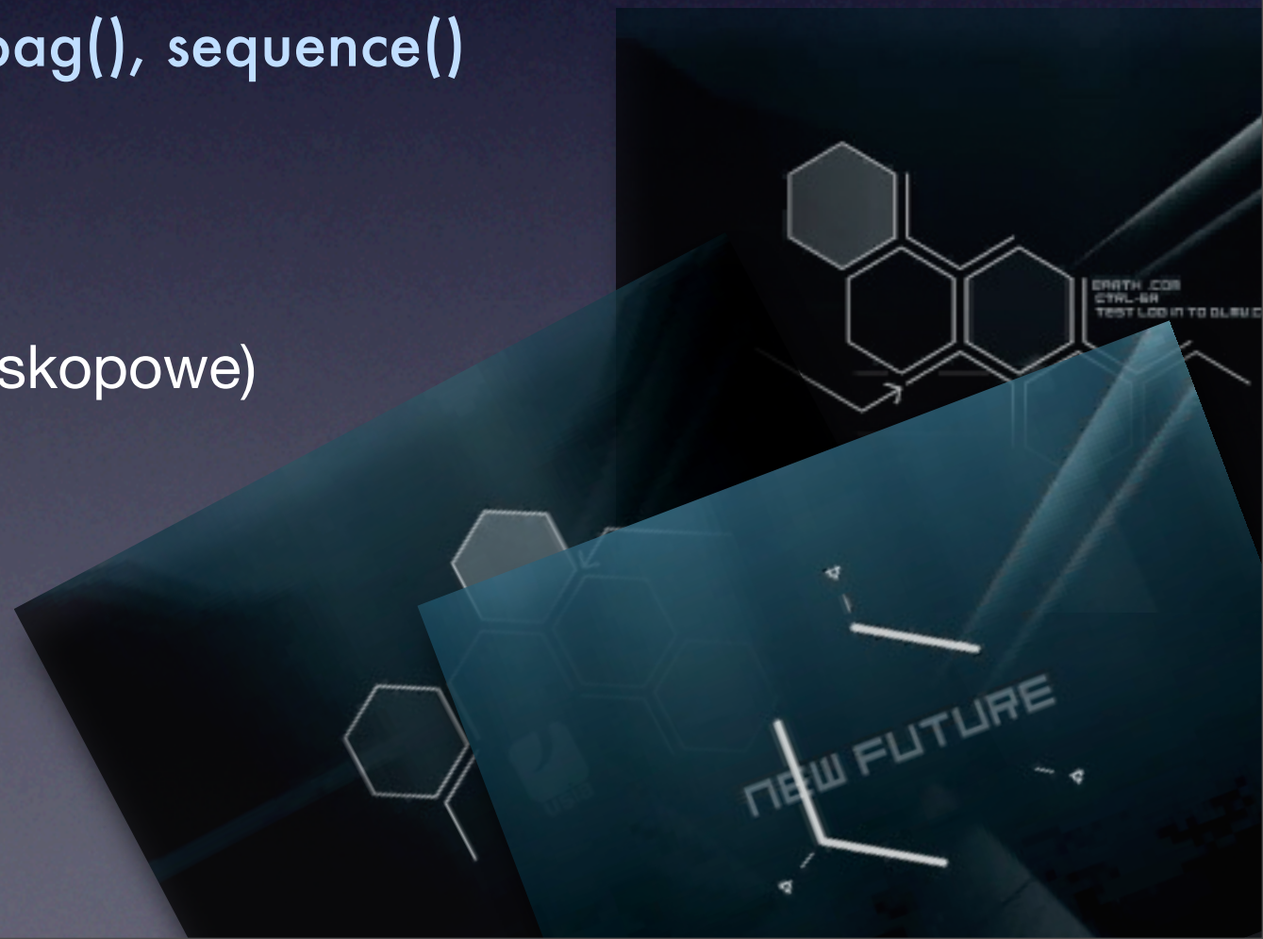
Ważniejsze operatory w SBQL

Ważniejsze operatory w SBQL

- operatory algebraiczne (makroskopowe i nie makroskopowe)
  - +, -, /, *, %, etc.
  - >, <, >=, <=, !=, etc.
  - or, and
  - union, intersect, minus, in
  - min(), max(), avg(), count(), bag(), sequence()
  - as, group as
  - ,

Ważniejsze operatory w SBQL

- operatory algebraiczne (makroskopowe i nie makroskopowe)
  - +, -, /, *, %, etc.
  - >, <, >=, <=, !=, etc.
  - or, and
  - union, intersect, minus, in
  - min(), max(), avg(), count(), bag(), sequence()
  - as, group as
  - ,
- operatory niealgebraiczne (makroskopowe)
  - where
  - .
  - join
  - order by
  - for all



Rodzaje rezultatów w SBQL

Rodzaje rezultatów w SBQL

- wartość atomowa
  np. 5, 3.5, “Ala ma kota”, true
- referencja
  np. i0
- struktura
  np. struct(1, 2, 3, 4)
- bag
  np. bag(1, 2, i0, “blabla”)
- sekwencja
  np. sequence(1, 2, 3, 4, 5)
- binder
  np. imie(i0), x(bag(1, 2, 3))

Stos rezultatów (QRES)

Stos rezultatów (QRES)

- Służy do tymczasowego przechowywania rezultatów zapytań

Stos rezultatów (QRES)

- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań

Stos rezultatów (QRES)

- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań
- Operacja **push** umieszcza nowy element na szczycie stosu

Stos rezultatów (QRES)

- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań
- Operacja **push** umieszcza nowy element na szczycie stosu
- Operacja **pop** usuwa element ze szczytu stosu

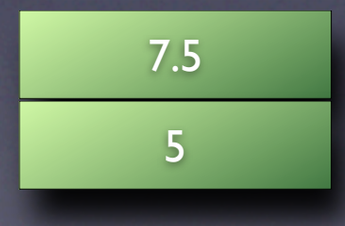
Stos rezultatów (QRES)

- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań
- Operacja **push** umieszcza nowy element na szczycie stosu
- Operacja **pop** usuwa element ze szczytu stosu

Przykład stosu QRES

Stos rezultatów (QRES)

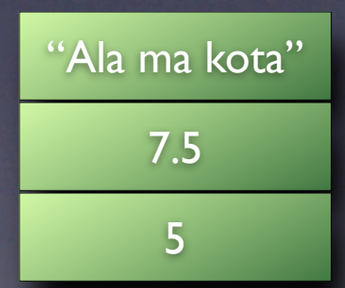
- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań
- Operacja **push** umieszcza nowy element na szczycie stosu
- Operacja **pop** usuwa element ze szczytu stosu



Przykład stosu QRES

Stos rezultatów (QRES)

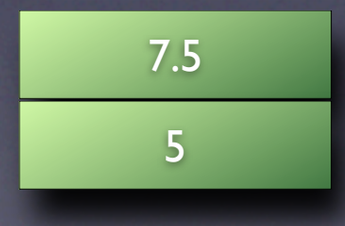
- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań
- Operacja **push** umieszcza nowy element na szczycie stosu
- Operacja **pop** usuwa element ze szczytu stosu



Przykład stosu QRES

Stos rezultatów (QRES)

- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań
- Operacja **push** umieszcza nowy element na szczycie stosu
- Operacja **pop** usuwa element ze szczytu stosu



Przykład stosu QRES

Proste zapytanie bez nazwy

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

2

Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

2 3

Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

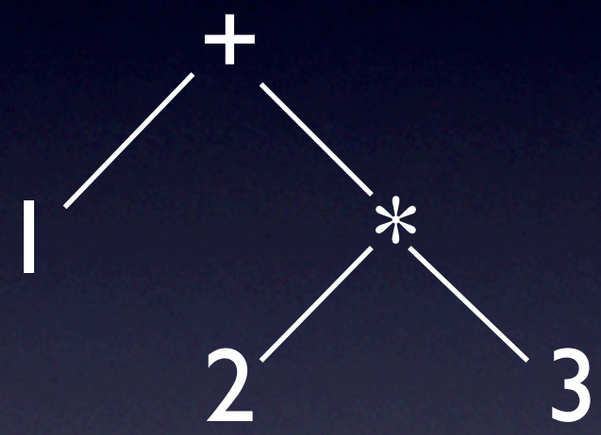
Proste zapytanie bez nazwy

1 + 2 * 3 - 4

Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

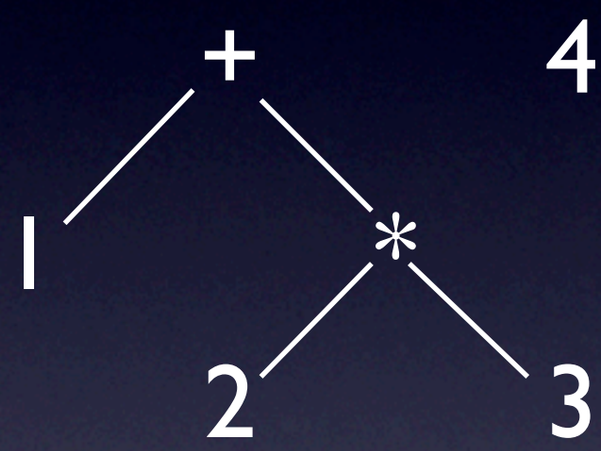
1 + 2 * 3 - 4



Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

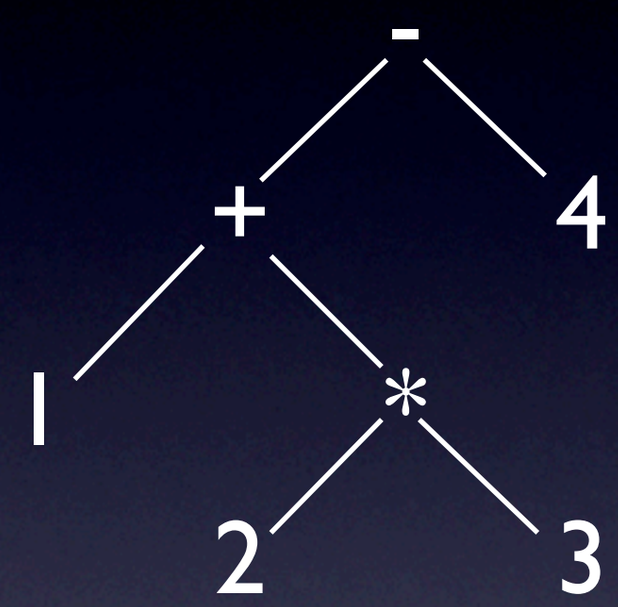
1 + 2 * 3 - 4



Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

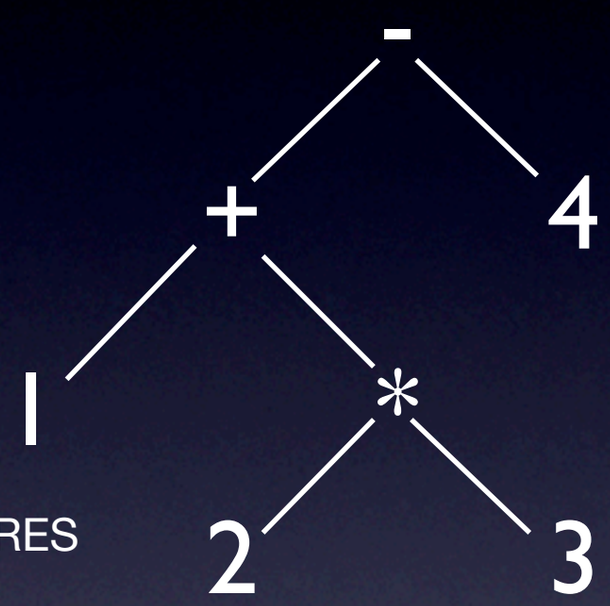


Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```

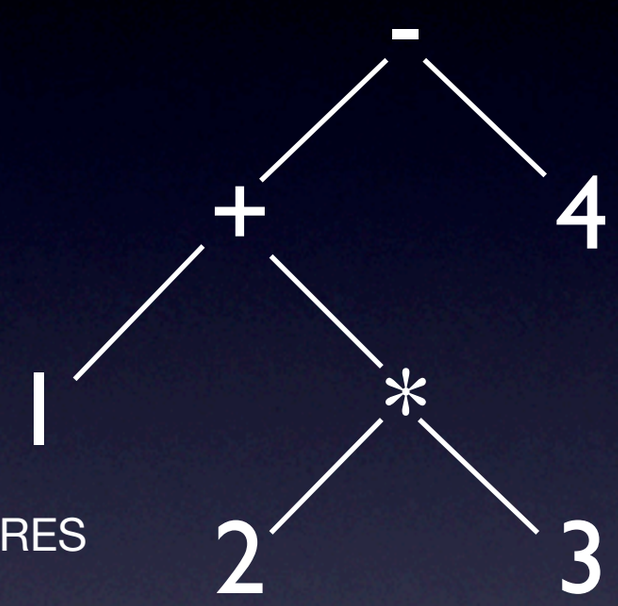


Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```



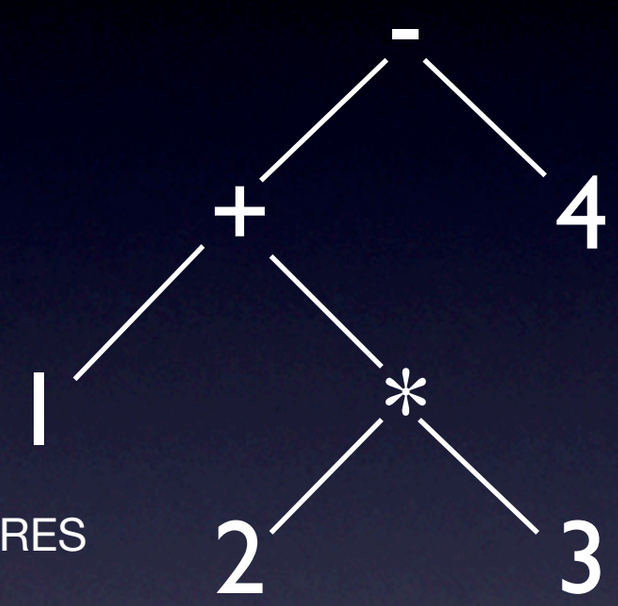
Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

1

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```

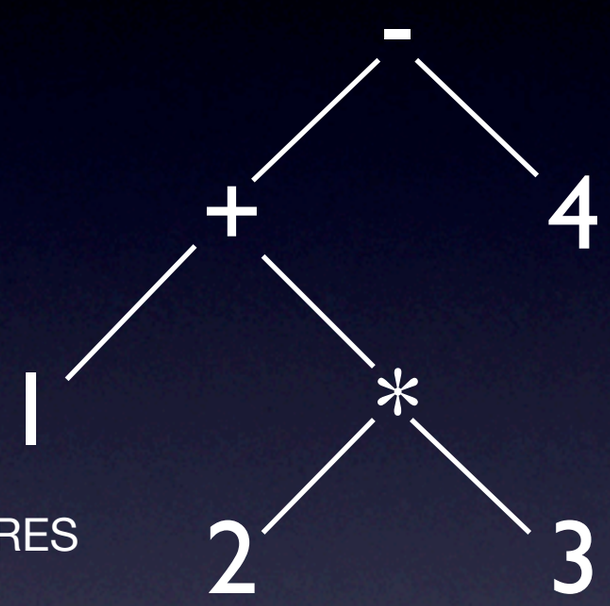


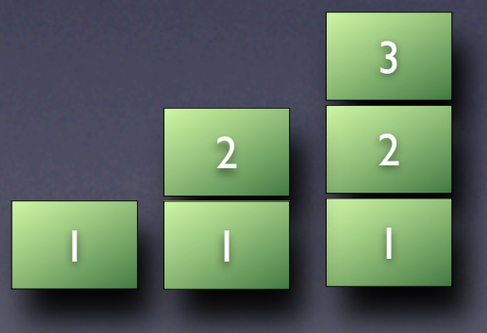
Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```



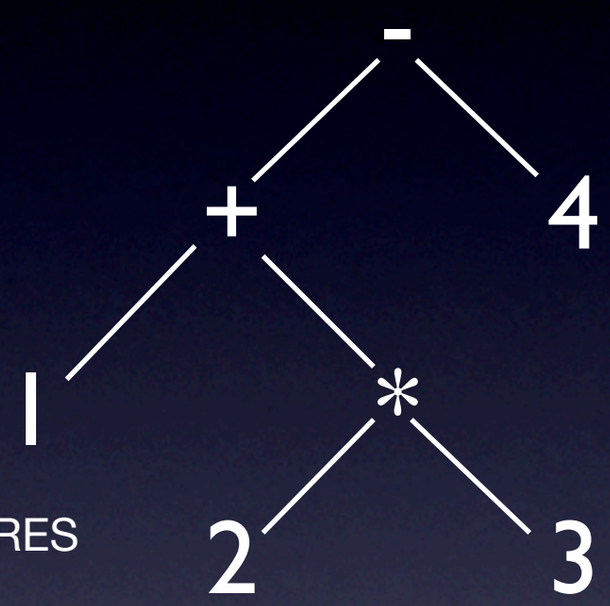


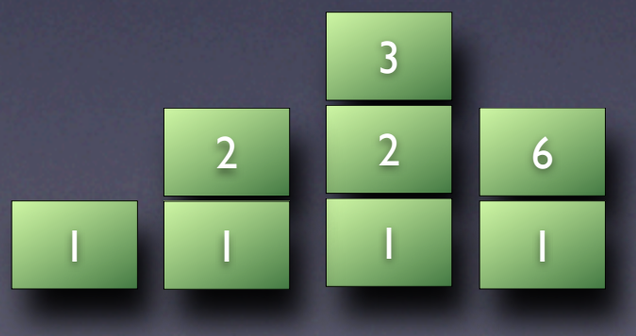
Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```



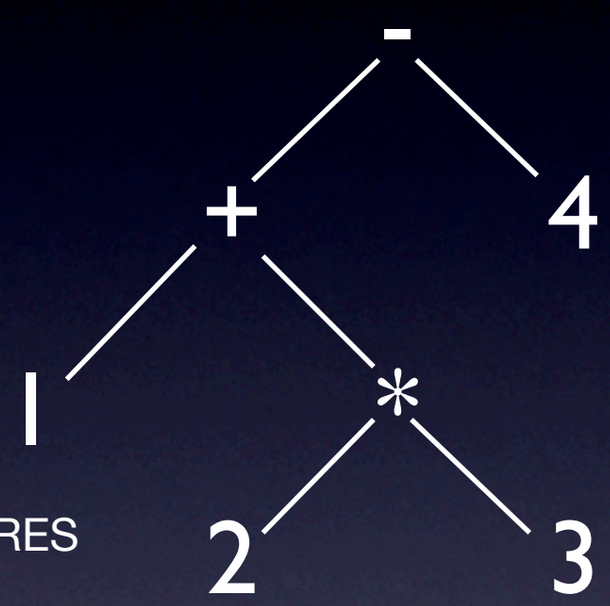


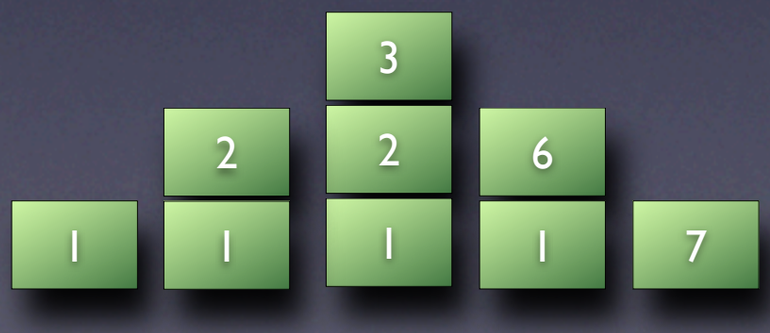
Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```



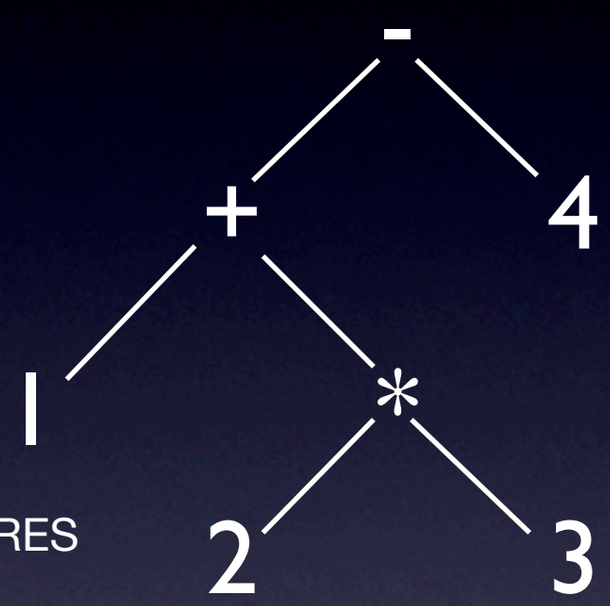


Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```

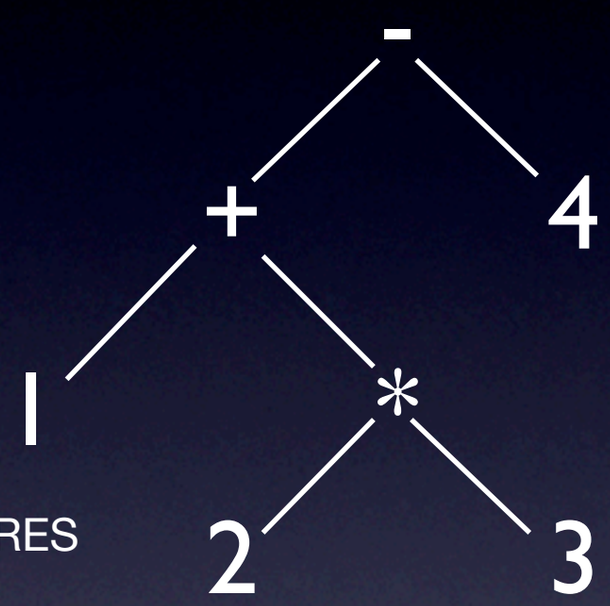


Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Proste zapytanie bez nazwy

1 + 2 * 3 - 4

```
tree_walk(node) {
 if (node is leaf)
  // wstaw wartość na QRES
 else if (node is operator) {
  tree_walk(node.left);
  tree_walk(node.right);
  // pobierz element(y) (zależy od operatora) z QRES
  // wykonaj operację związaną z operatorem
  // odłóż wynik na QRES
}
```



Jeśli w zapytaniu nie ma nazw,
to do ewaluacji wystarczy stos QRES

Stos środowiskowy (ENVS)

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych).
  Trafiają do niej bindery do obiektów korzeniowych bazy danych

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu
- Operacja **bind** dokonuje wiązania nazw

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu
- Operacja **bind** dokonuje wiązania nazw
- Operacja **push** zazwyczaj wywoływana jest razem z operacją **nested**

Stos środowiskowy (ENVS)

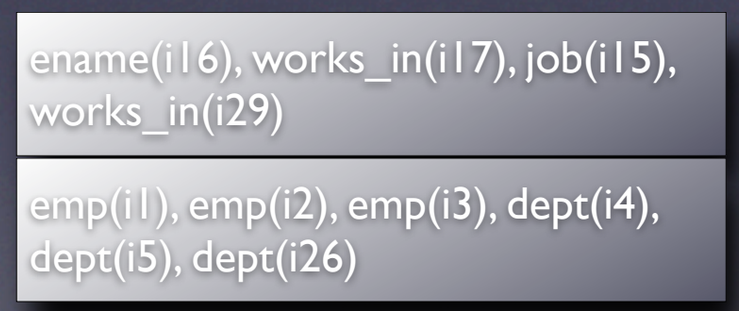
- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu
- Operacja **bind** dokonuje wiązania nazw
- Operacja **push** zazwyczaj wywoływana jest razem z operacją **nested**

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Przykład stosu ENVS

Stos środowiskowy (ENVS)

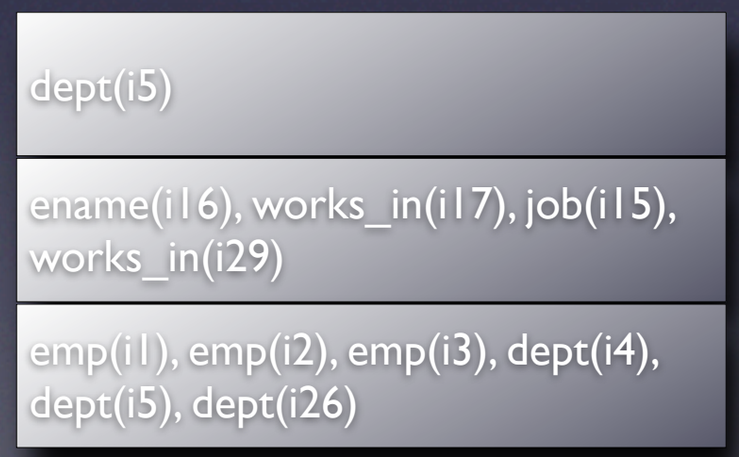
- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu
- Operacja **bind** dokonuje wiązania nazw
- Operacja **push** zazwyczaj wywoływana jest razem z operacją **nested**



Przykład stosu ENVS

Stos środowiskowy (ENVS)

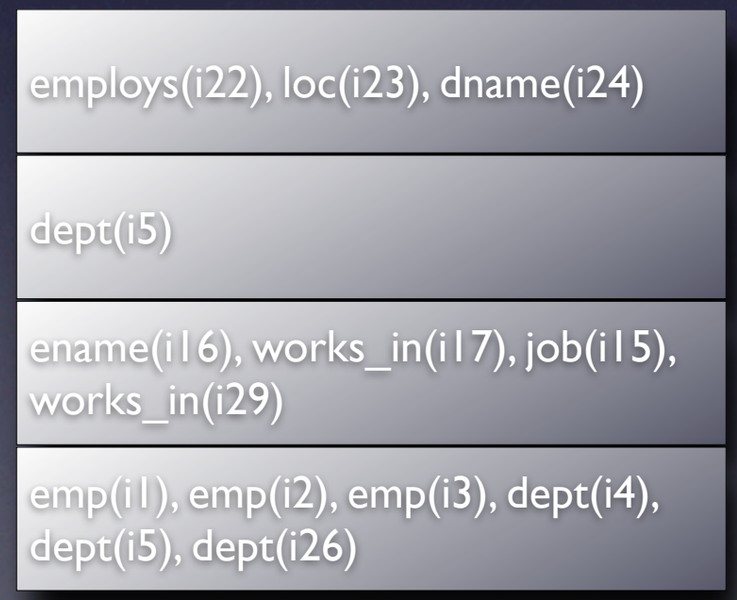
- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu
- Operacja **bind** dokonuje wiązania nazw
- Operacja **push** zazwyczaj wywoływana jest razem z operacją **nested**



Przykład stosu ENVS

Stos środowiskowy (ENVS)

- Służy do zamieniania nazw występujących w tekscie programu na konkretne byty programistyczne (wiązanie nazw)
- Składa się z sekcji, a każda sekcja składa się z binderów
- Pierwsza sekcja (sekcja bazowa) inicjalizowana jest automatycznie zawartością bazy danych). Trafiają do niej bindery do obiektów korzeniowych bazy danych
- Operacja **push** umieszcza nową sekcję na szczycie stosu
- Operacje **pop** usuwa sekcję ze szczytu stosu
- Operacja **bind** dokonuje wiązania nazw
- Operacja **push** zazwyczaj wywoływana jest razem z operacją **nested**



Przykład stosu ENVS

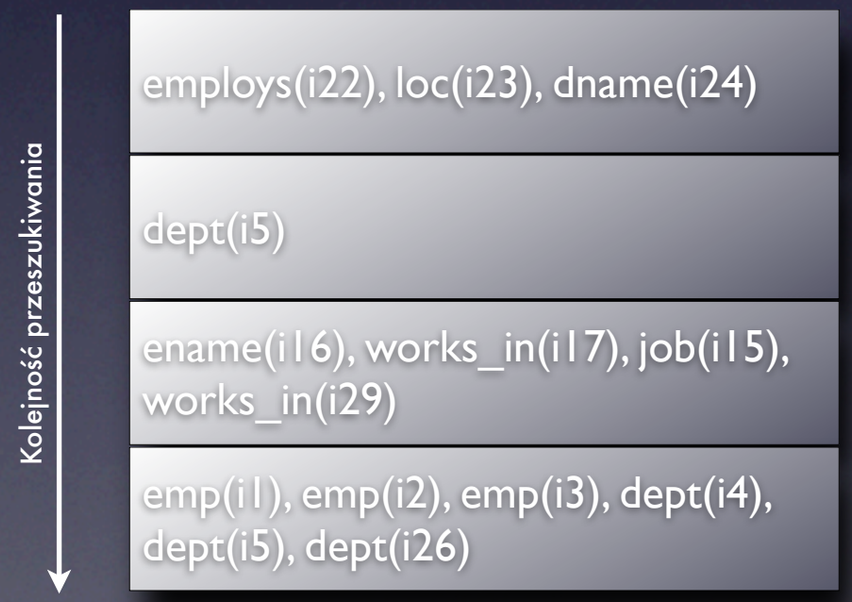
Wiązanie nazw

Wiązanie nazw

Operacja **bind** pobiera jako argument nazwę, a następnie przechodzi ENVS od góry w dół poszukując tej nazwy. Przeszukiwanie przerywane jest w pierwszej sekcji, w której nazwa zostanie znaleziona.

Wiązanie nazw

Operacja **bind** pobiera jako argument nazwę, a następnie przechodzi ENVS od góry w dół poszukując tej nazwy. Przeszukiwanie przerywane jest w pierwszej sekcji, w której nazwa zostanie znaleziona.

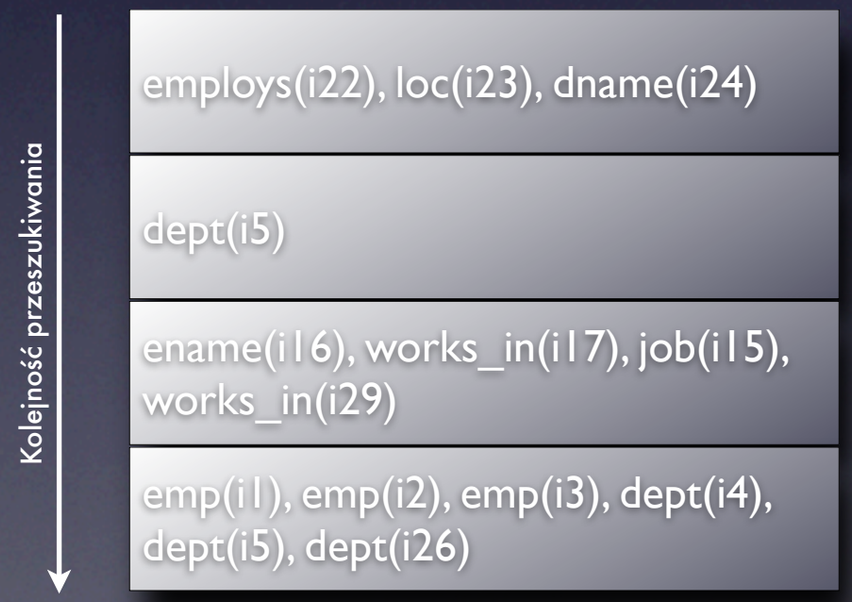


Wiązanie nazw

Operacja **bind** pobiera jako argument nazwę, a następnie przechodzi ENVS od góry w dół poszukując tej nazwy. Przeszukiwanie przerywane jest w pierwszej sekcji, w której nazwa zostanie znaleziona.

Przykłady:

bind(”emp”) = bag(i1, i2, i3)
bind(”dept”) = bag(i5)
bind(”works_in”) = bag(i17, i29)
bind(”loc”) = bag(i23)



Proste zapytanie z nazwą

Proste zapytanie z nazwą

emp

Proste zapytanie z nazwą

emp

Ewaluacja elementarnych zapytań będących nazwami polega na wywołaniu operacji **bind**

Proste zapytanie z nazwą

emp

Ewaluacja elementarnych zapytań będących nazwami polega na wywołaniu operacji **bind**

Zatem jeśli w dowolnym zapytaniu występuje nazwa, to do ewaluacji potrzebny jest stos QRES i stos ENVS

Proste zapytanie z nazwą

emp

Ewaluacja elementarnych zapytań będących nazwami polega na wywołaniu operacji **bind**

Zatem jeśli w dowolnym zapytaniu występuje nazwa, to do ewaluacji potrzebny jest stos QRES i stos ENVS

Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), eept(i4), dept(i5), dept(i26)

Proste zapytanie z nazwą

emp

Ewaluacja elementarnych zapytań będących nazwami polega na wywołaniu operacji **bind**

Zatem jeśli w dowolnym zapytaniu występuje nazwa, to do ewaluacji potrzebny jest stos QRES i stos ENVS

Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), eept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”Emp”).
Wynik na QRES. ENVS pozostaje bez zmian.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i1, i2, i3)

Proste zapytanie z nazwą

emp

Ewaluacja elementarnych zapytań będących nazwami polega na wywołaniu operacji **bind**

Zatem jeśli w dowolnym zapytaniu występuje nazwa, to do ewaluacji potrzebny jest stos QRES i stos ENVS

Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), eept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”Emp”).
Wynik na QRES. ENVS pozostaje bez zmian.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i1, i2, i3)

Inne zapytania nie są już takie proste ...

Wyznaczanie wnętrza obiektów

Wyznaczanie wnętrza obiektów

Operacja **nested** pobiera jako argument rezultat zapytania.
W zależności od rodzaju argumentu wynikiem jest:

Wyznaczanie wnętrza obiektów

Operacja **nested** pobiera jako argument rezultat zapytania.
W zależności od rodzaju argumentu wynikiem jest:

- Dla wartości atomowej: pusty zbiór
- Dla referencji do obiektu prostego: pusty zbiór
- Dla referencji do obiektu złożonego:
  zbiór zawierający bindery z nazwami
  identyfikatorami podobiektów tego obiektu
- Dla referencji do obiektu referencyjnego:
  zbiór zawierający binder z nazwą
  i identyfikatorem obiektu wskazywanego
- Dla bindera: ten sam binder
- Dla struktury: suma operacji nested wszystkich elementów struktury
- Dla pozostałych rezultatów: pusty zbiór

Wyznaczanie wnętrza obiektów

Operacja **nested** pobiera jako argument rezultat zapytania.
W zależności od rodzaju argumentu wynikiem jest:

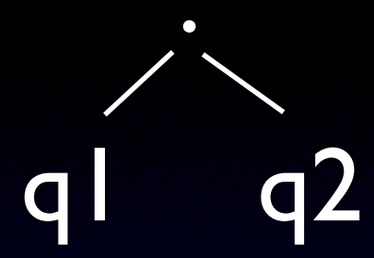
- Dla wartości atomowej: pusty zbiór
- Dla referencji do obiektu prostego: pusty zbiór
- Dla referencji do obiektu złożonego:
  zbiór zawierający bindery z nazwami
  identyfikatorami podobiektów tego obiektu
- Dla referencji do obiektu referencyjnego:
  zbiór zawierający binder z nazwą
  i identyfikatorem obiektu wskazywanego
- Dla bindera: ten sam binder
- Dla struktury: suma operacji nested wszystkich elementów struktury
- Dla pozostałych rezultatów: pusty zbiór

```
nested(5) = { }
nested(i4) = { loc(i6),
               dname(i7),
               employs(i8),
               employs(i9) }
nested(i8) = { emp(i3) }
nested(struct(i26, i29)) =
               { loc(i27),
                 dname(i28),
                 dept(i26) }
```

Operator .

Operator .

q1.q2



Operator .

q1.q2

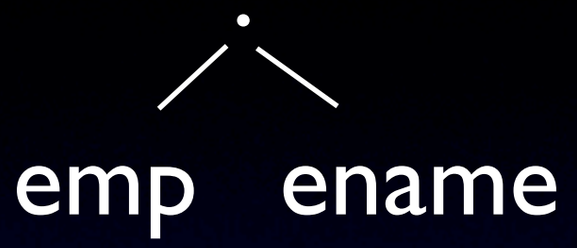
- Zainicjalizować dotres = bag()
- Wykonać **eval**(q1) i zrobić q1res = QRES.POP()
- Dla każdego elementu x ∈ q1res wykonać:
  - Utworzyć nową sekcję na ENVS
  - Wykonać nested(x). Wynik wprowadzić do sekcji utworzonej w poprzednim kroku
  - Wykonać eval(q2) i zrobić q2res = QRES.POP()
  - Dodać q2res do dotres (dotres ∪ q2res)
  - Zamknąć sekcję ENVS
- Zrobić QRES.PUSH(dotres)

Przykład

Przykład

emp.ename

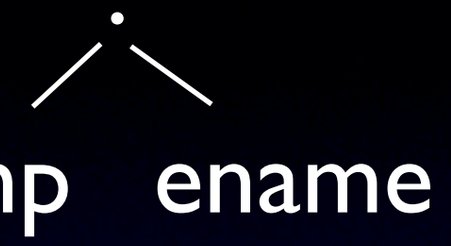
Zapytanie powinno nam zwrócić referencje do wszystkich nazwisk pracowników



Przykład

emp.ename

Zapytanie powinno nam zwrócić referencje do wszystkich nazwisk pracowników



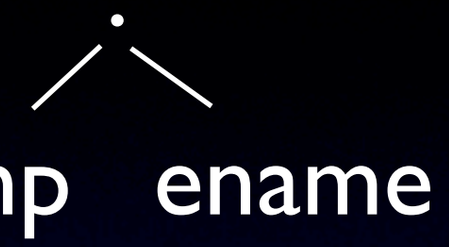
Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Przykład

emp.ename

Zapytanie powinno nam zwrócić referencje do wszystkich nazwisk pracowników



Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”emp”).
Wynik na QRES. ENVS pozostaje bez zmian.

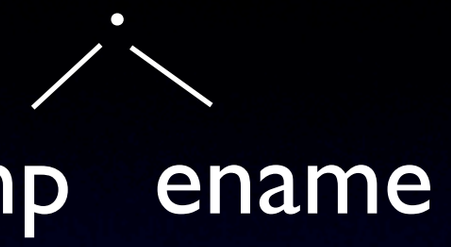
emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i1, i2, i3)

Przykład

emp.ename

Zapytanie powinno nam zwrócić referencje do wszystkich nazwisk pracowników



Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”emp”).
Wynik na QRES. ENVS pozostaje bez zmian.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i1, i2, i3)

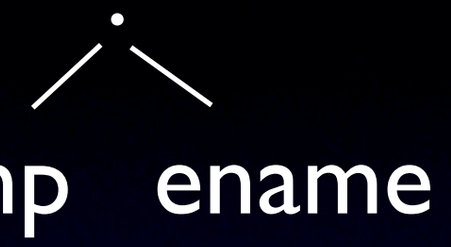
Robimy QRES.POP().
ENVS pozostaje bez zmian. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Przykład

emp.ename

Zapytanie powinno nam zwrócić referencje do wszystkich nazwisk pracowników



Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”emp”).
Wynik na QRES. ENVS pozostaje bez zmian.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i1, i2, i3)

Robimy QRES.POP().
ENVS pozostaje bez zmian. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Robimy nested(i1).
ENVS dostaje nową sekcję. QRES bez zmian.

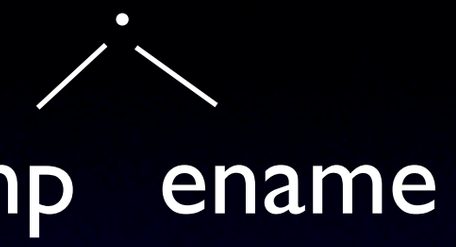
ename(i16), works_in(i17), job(i15), works_in(i29)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Przykład

emp.ename

Zapytanie powinno nam zwrócić referencje do wszystkich nazwisk pracowników



Inicjalizujemy ENVS i QRES.
ENVS z binderami obiektów korzeniowych. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”emp”).
Wynik na QRES. ENVS pozostaje bez zmian.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i1, i2, i3)

Robimy QRES.POP().
ENVS pozostaje bez zmian. QRES pusty.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

ename(i16), works_in(i17), job(i15), works_in(i29)

Robimy nested(i1).
ENVS dostaje nową sekcję. QRES bez zmian.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

ename(i16), works_in(i17), job(i15), works_in(i29)

Wyliczamy **bind**(”ename”).
Wynik wrzucamy na QRES.

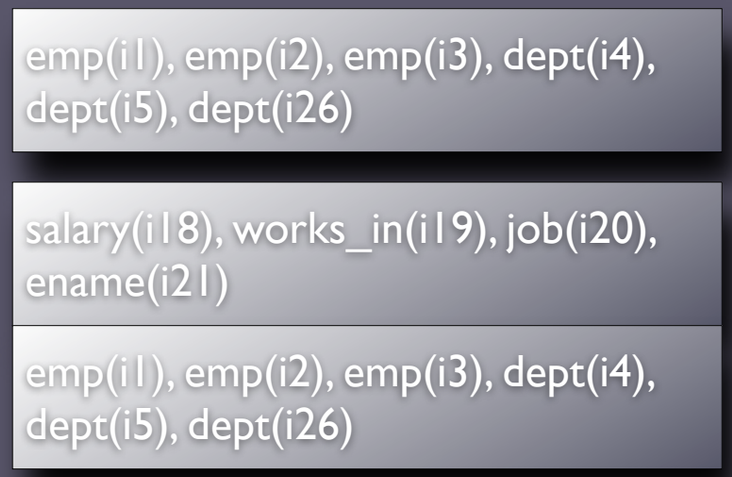
emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i16)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES trafia do resdot, więc resdot = bag{i16}.

emp(i1), emp(i2), emp(i3), dept(i4),
dept(i5), dept(i26)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES trafia do resdot, więc resdot = bag{i16}.

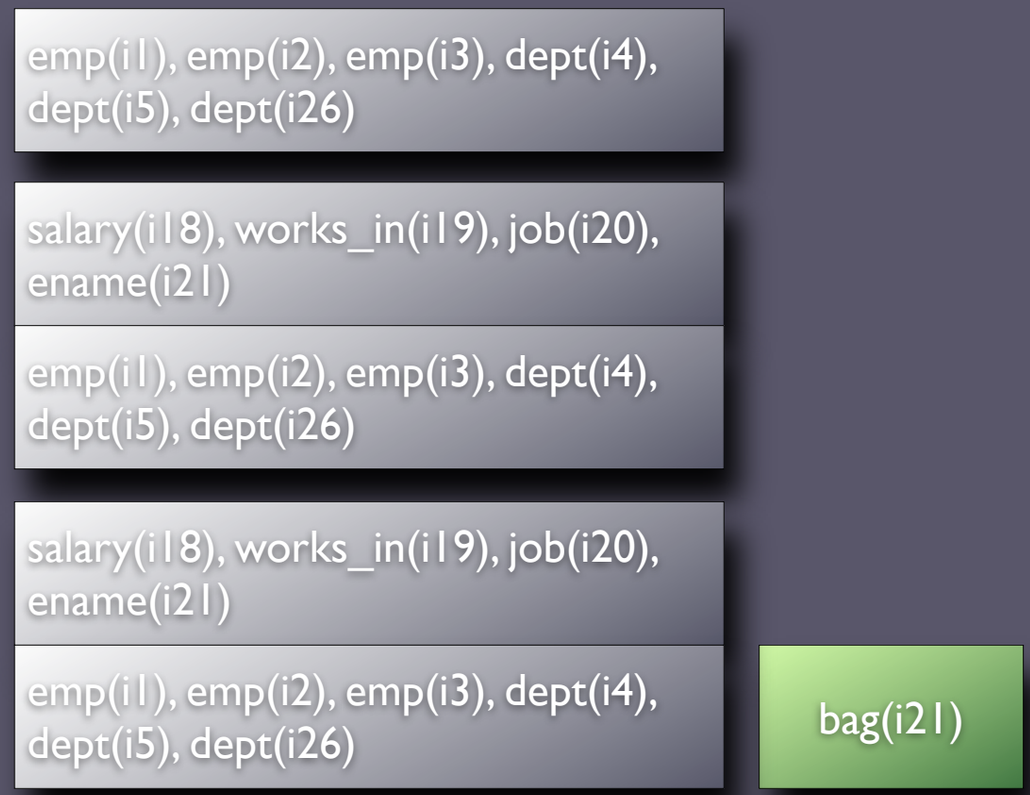


Robimy nested(i2).
ENVS dostaje nową sekcję. QRES bez zmian.

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES trafia do resdot, więc resdot = bag{i16}.

Robimy nested(i2).
ENVS dostaje nową sekcję. QRES bez zmian.

Wyliczamy **bind**(”ename”).
Wynik wrzucamy na QRES.



Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES trafia do resdot, więc resdot = bag{i16}.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Robimy nested(i2).
ENVS dostaje nową sekcję. QRES bez zmian.

salary(i18), works_in(i19), job(i20), ename(i21)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”ename”).
Wynik wrzucamy na QRES.

salary(i18), works_in(i19), job(i20), ename(i21)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i21)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES do resdot, więc resdot = bag{i16, i21}.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES trafia do resdot, więc resdot = bag{i16}.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Robimy nested(i2).
ENVS dostaje nową sekcję. QRES bez zmian.

salary(i18), works_in(i19), job(i20), ename(i21)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”ename”).
Wynik wrzucamy na QRES.

salary(i18), works_in(i19), job(i20), ename(i21)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i21)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES do resdot, więc resdot = bag{i16, i21}.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Robimy nested(i3).
ENVS dostaje nową sekcję. QRES bez zmian.

salary(i10), job(i11), ename(i12), works_in(i13)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES trafia do resdot, więc resdot = bag{i16}.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Robimy nested(i2).
ENVS dostaje nową sekcję. QRES bez zmian.

salary(i18), works_in(i19), job(i20), ename(i21)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”ename”).
Wynik wrzucamy na QRES.

salary(i18), works_in(i19), job(i20), ename(i21)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i21)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES do resdot, więc resdot = bag{i16, i21}.

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Robimy nested(i3).
ENVS dostaje nową sekcję. QRES bez zmian.

salary(i10), job(i11), ename(i12), works_in(i13)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

Wyliczamy **bind**(”ename”).
Wynik wrzucamy na QRES.

salary(i10), job(i11), ename(i12), works_in(i13)

emp(i1), emp(i2), emp(i3), dept(i4), dept(i5), dept(i26)

bag(i12)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES do resdot, resdot = bag{i16, i21, i12}.

emp(i1), emp(i2), emp(i3), dept (i4), dept(i5), dept(i26)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES do resdot, resdot = bag{i16, i21, i12}.

emp(i1), emp(i2), emp(i3), dept (i4), dept(i5), dept(i26)

Wrzucamy resdot na QRES.
Koniec wykonywania zapytania.

emp(i1), emp(i2), emp(i3), dept (i4), dept(i5), dept(i26)

bag(i16, i21, i12)

Usuwamy sekcję z ENVS i wartość z QRES.
Wartość z QRES do resdot, resdot = bag{i16, i21, i12}.

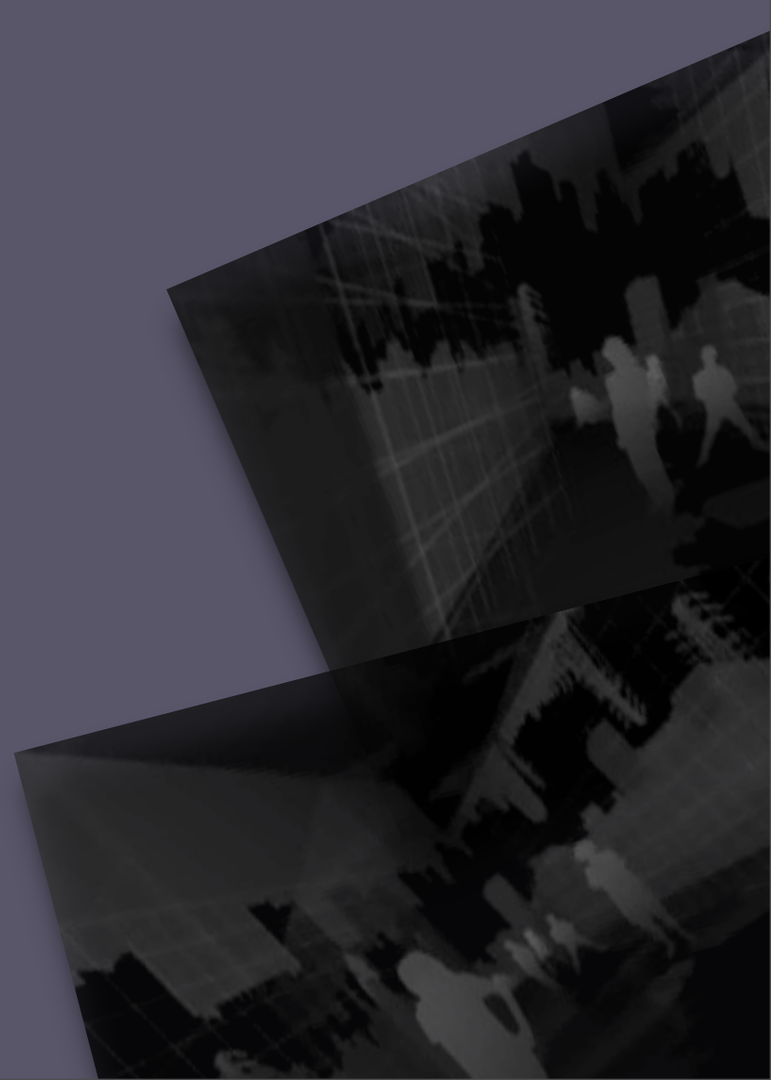
Wrzucamy resdot na QRES.
Koniec wykonywania zapytania.

emp(i1), emp(i2), emp(i3), dept (i4), dept(i5), dept(i26)

emp(i1), emp(i2), emp(i3), dept (i4), dept(i5), dept(i26)

bag(i16, i21, i12)

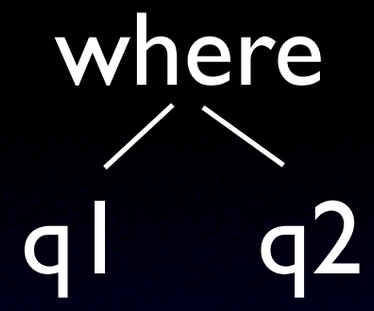
W ten sposób mamy nazwiska wszystkich pracowników



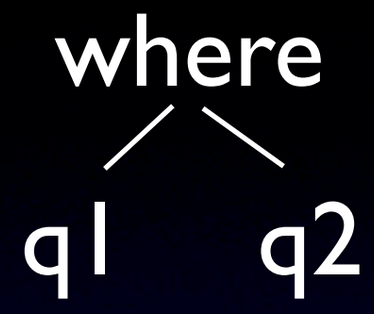
Operator where

Operator where

q1 where q2



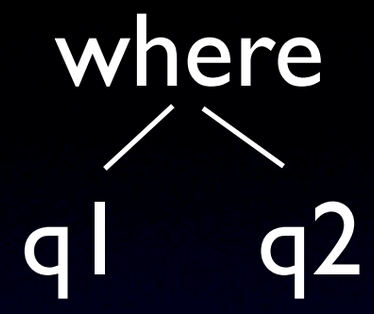
Operator where



q1 where q2

- Zainicjalizować whereres = bag()
- Wykonać **eval**(q1) i zrobić q1res = QRES.POP()
- Dla każdego elementu x ∈ q1res wykonać:
  - Utworzyć nową sekcję na ENVS
  - Wykonać nested(x). Wynik wprowadzić do sekcji utworzonej w poprzednim kroku
  - Wykonać eval(q2) i zrobić q2res = QRES.POP()
  - Jeśli q2res == true, wtedy dodać x do whereres
  - Zamknąć sekcję ENVS
- Zrobić QRES.PUSH(whereres)

Operator where



q1 where q2

- Zainicjalizować whereres = bag()
- Wykonać **eval**(q1) i zrobić q1res = QRES.POP()
- Dla każdego elementu x ∈ q1res wykonać:
  - Utworzyć nową sekcję na ENVS
  - Wykonać nested(x). Wynik wprowadzić do sekcji utworzonej w poprzednim kroku
  - Wykonać eval(q2) i zrobić q2res = QRES.POP()
  - Jeśli q2res == true, wtedy dodać x do whereres
  - Zamknąć sekcję ENVS
- Zrobić QRES.PUSH(whereres)

Reszta podobnie jak w przypadku operatora . ...

Ćwiczenia