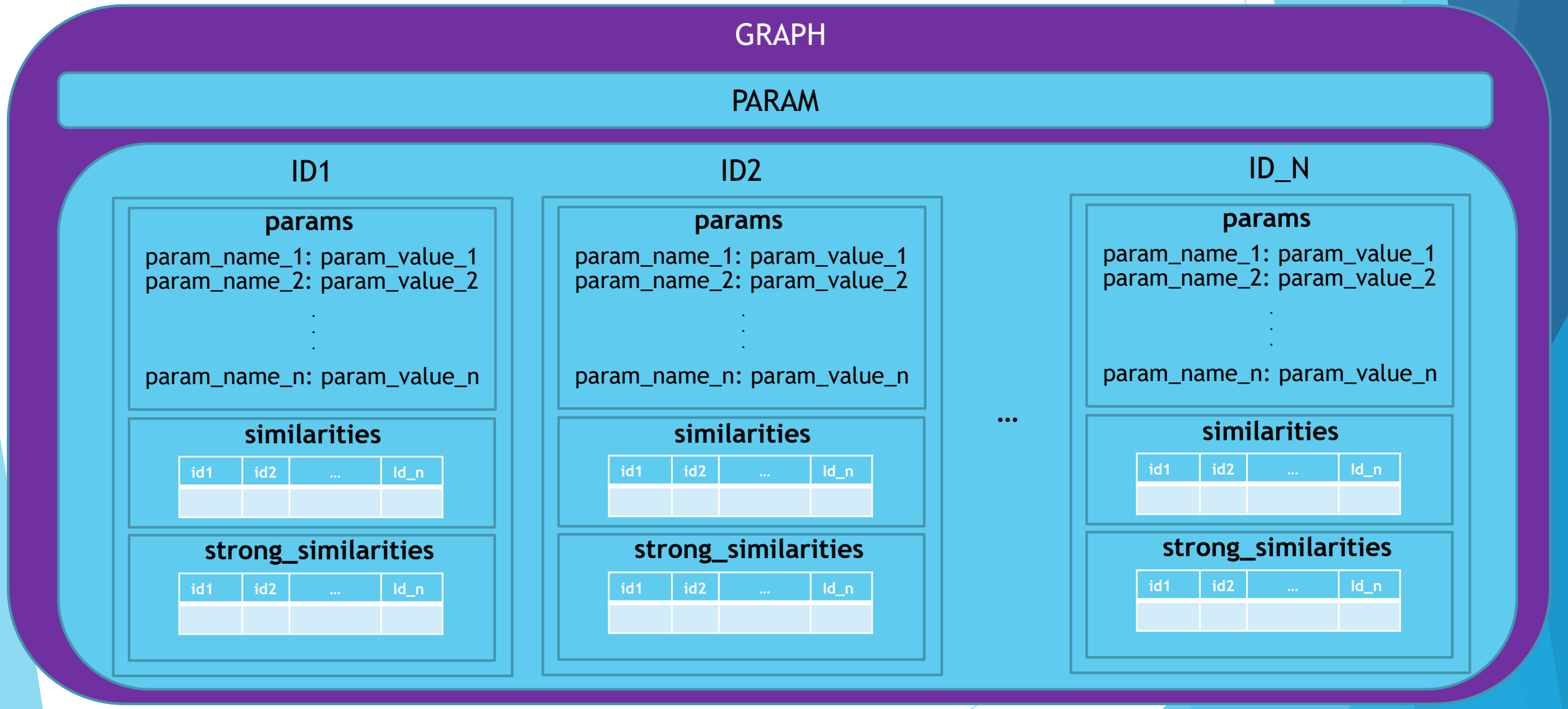
Asocjacyjna reprezentacja danych i wnioskowanie

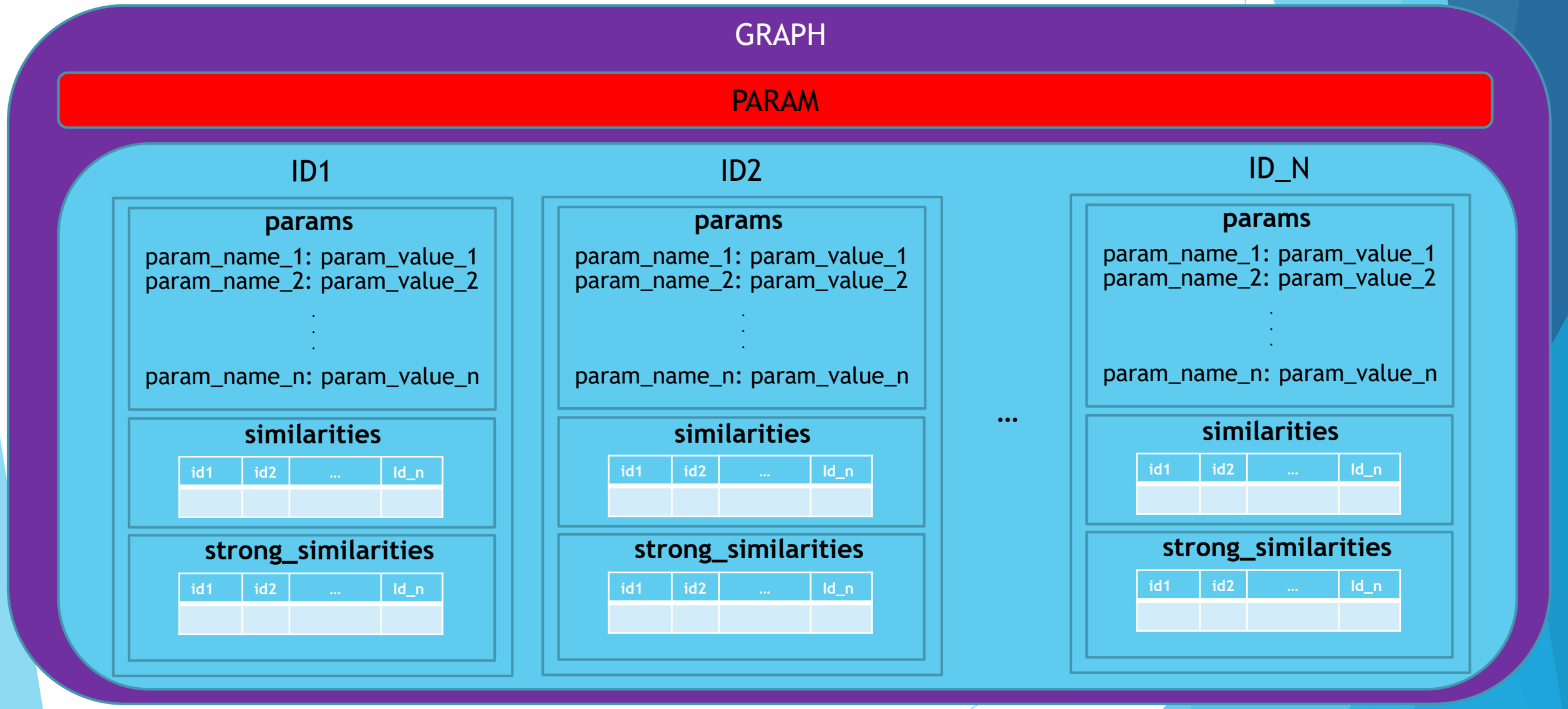
Wykorzystane technologie

- JetBrains PyCharm 5.0.4
- Python 3.5

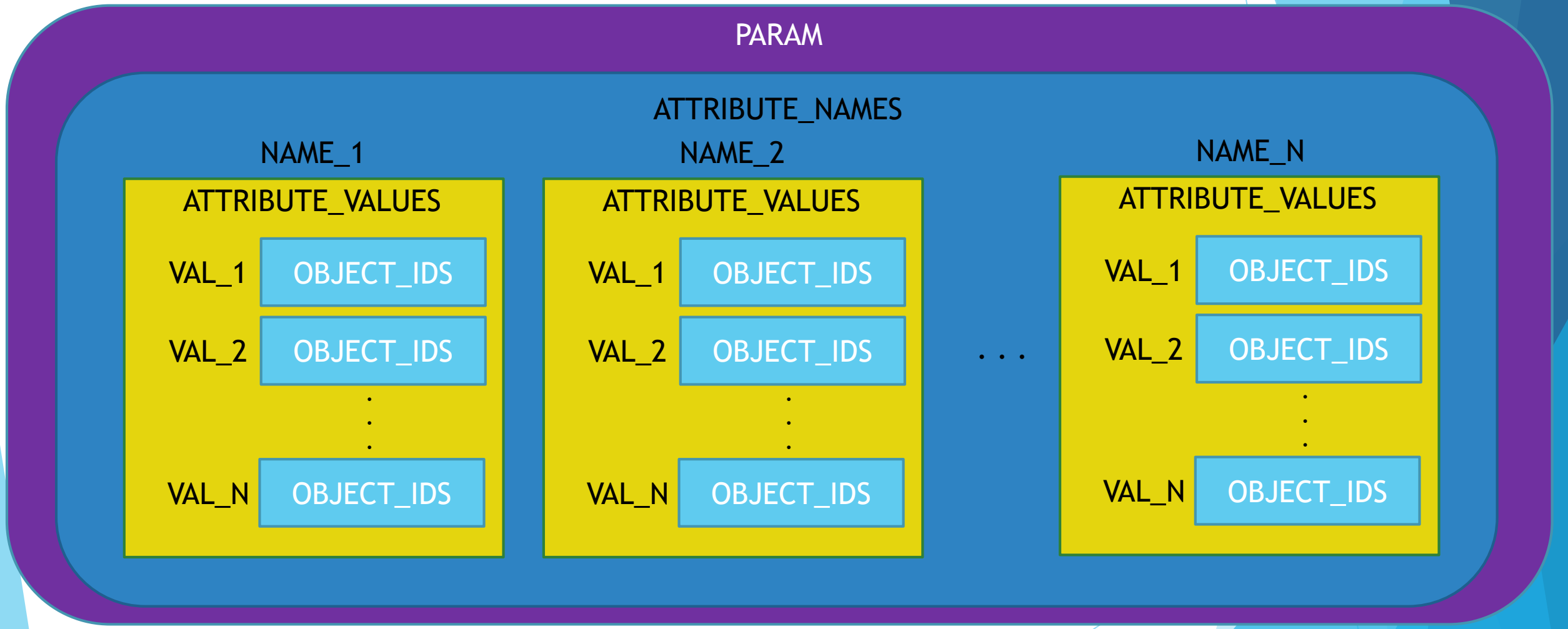
Struktura drzewa



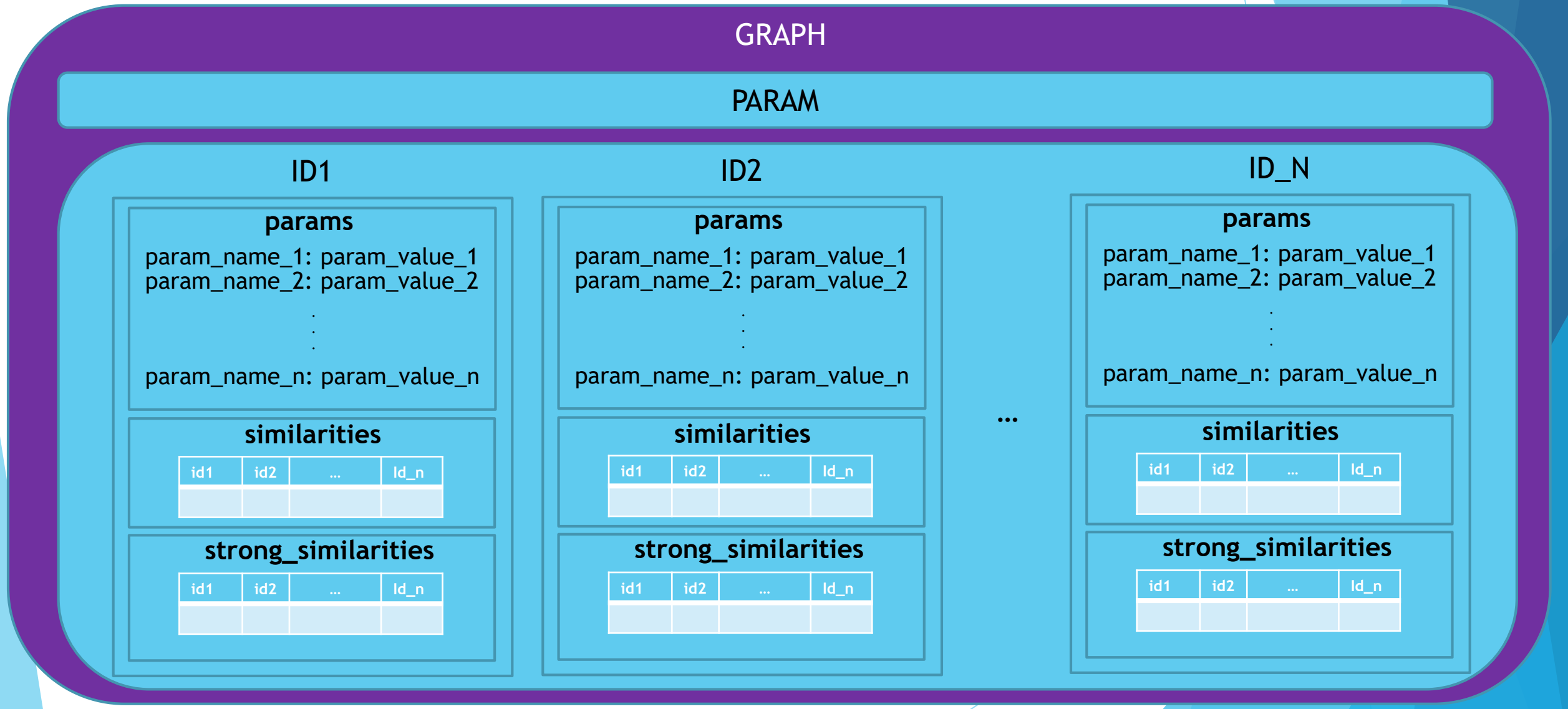
Struktura drzewa



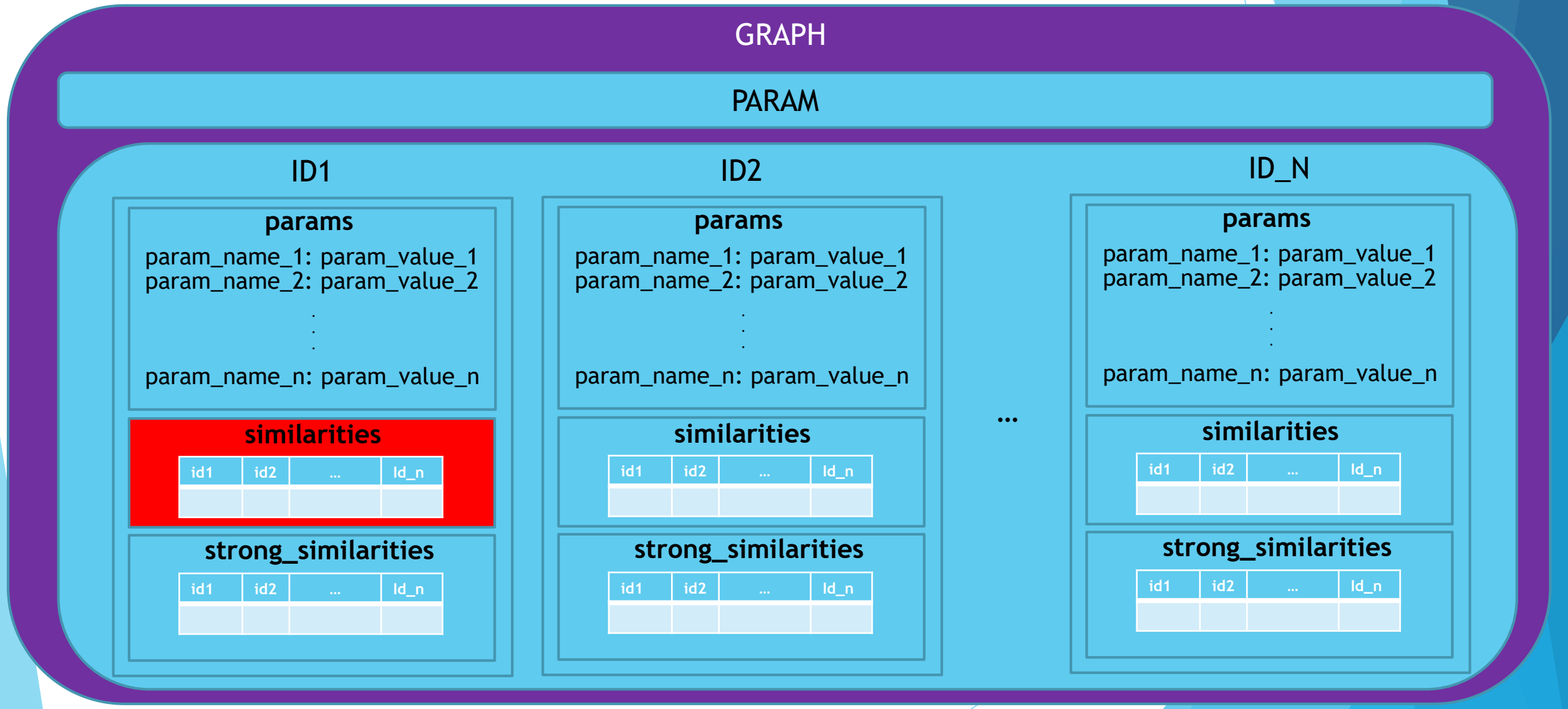
Struktura drzewa



Struktura drzewa



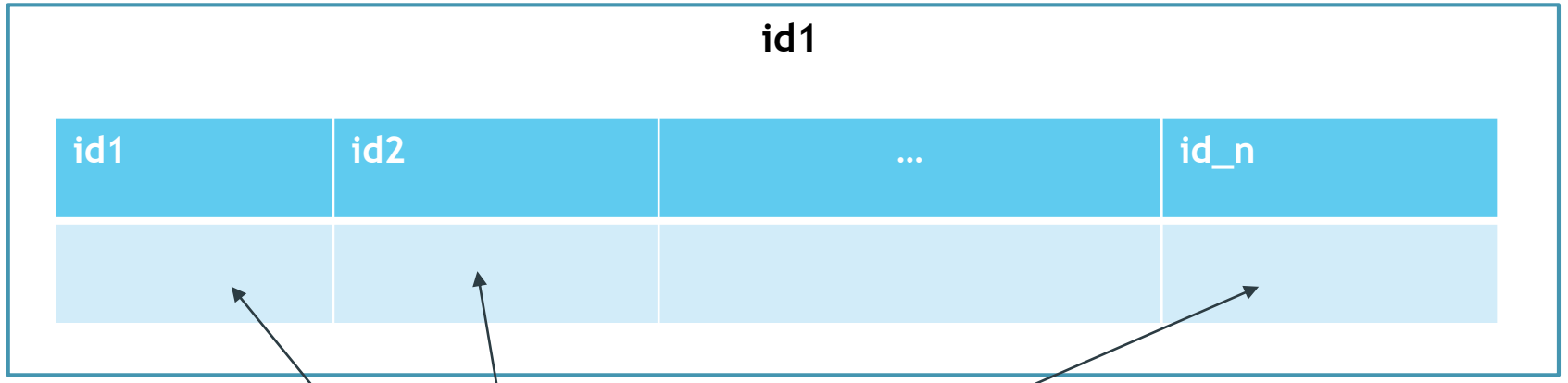
Struktura drzewa



Similarities – Wagi podobienstw

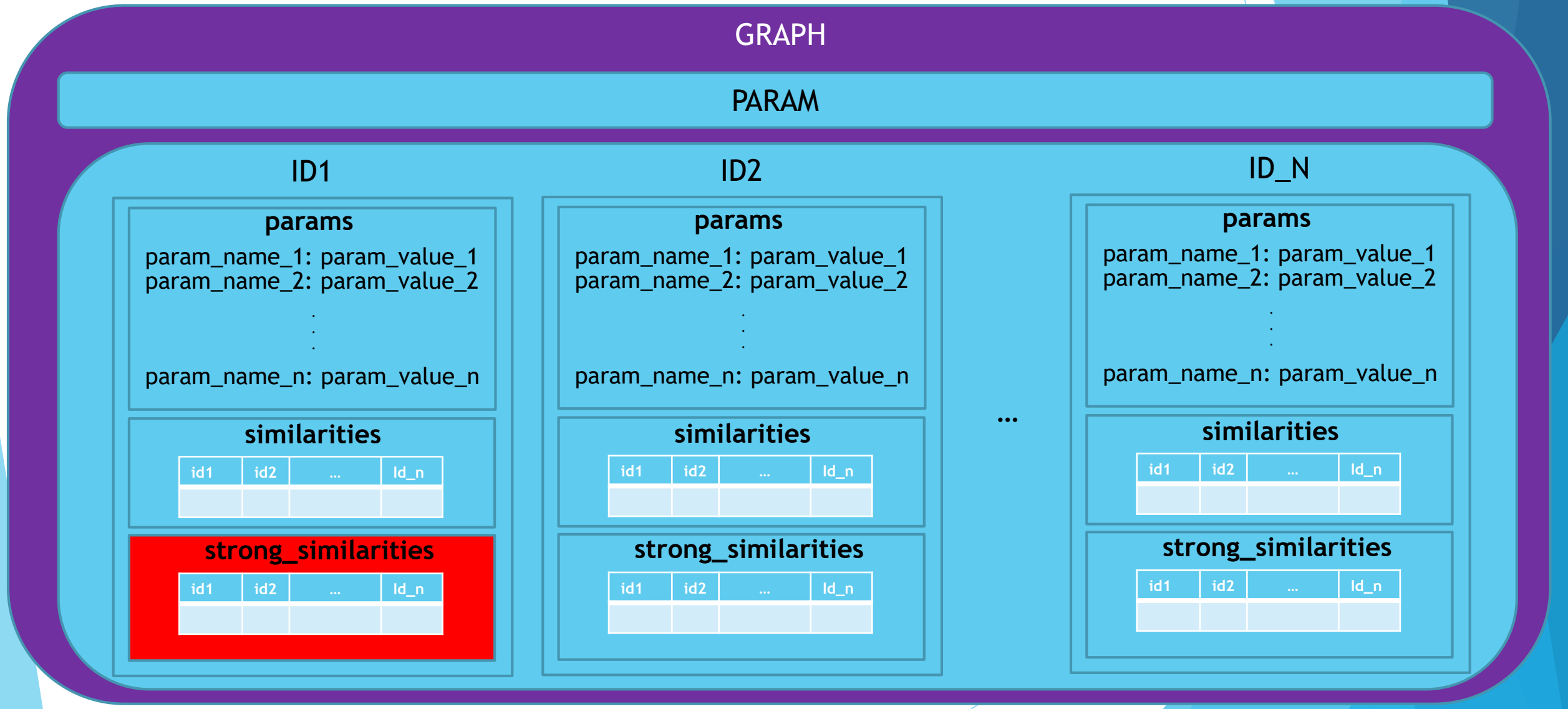
Każdy obiekt ma tablicę podobieństw o rozmiarze równym ilości obiektów w bazie

Każdy element tablicy odpowiada wartości podobieństwa danego obiektu i obiektu o id = indeksowi tablicy



$$w_{a,b} = \sum_{i=0}^{n} (1 - \frac{|a-b|}{MAX_{P_i} - MIN_{P_i}})$$

Struktura drzewa



Strong similarities – Wagi silnych podobieństw

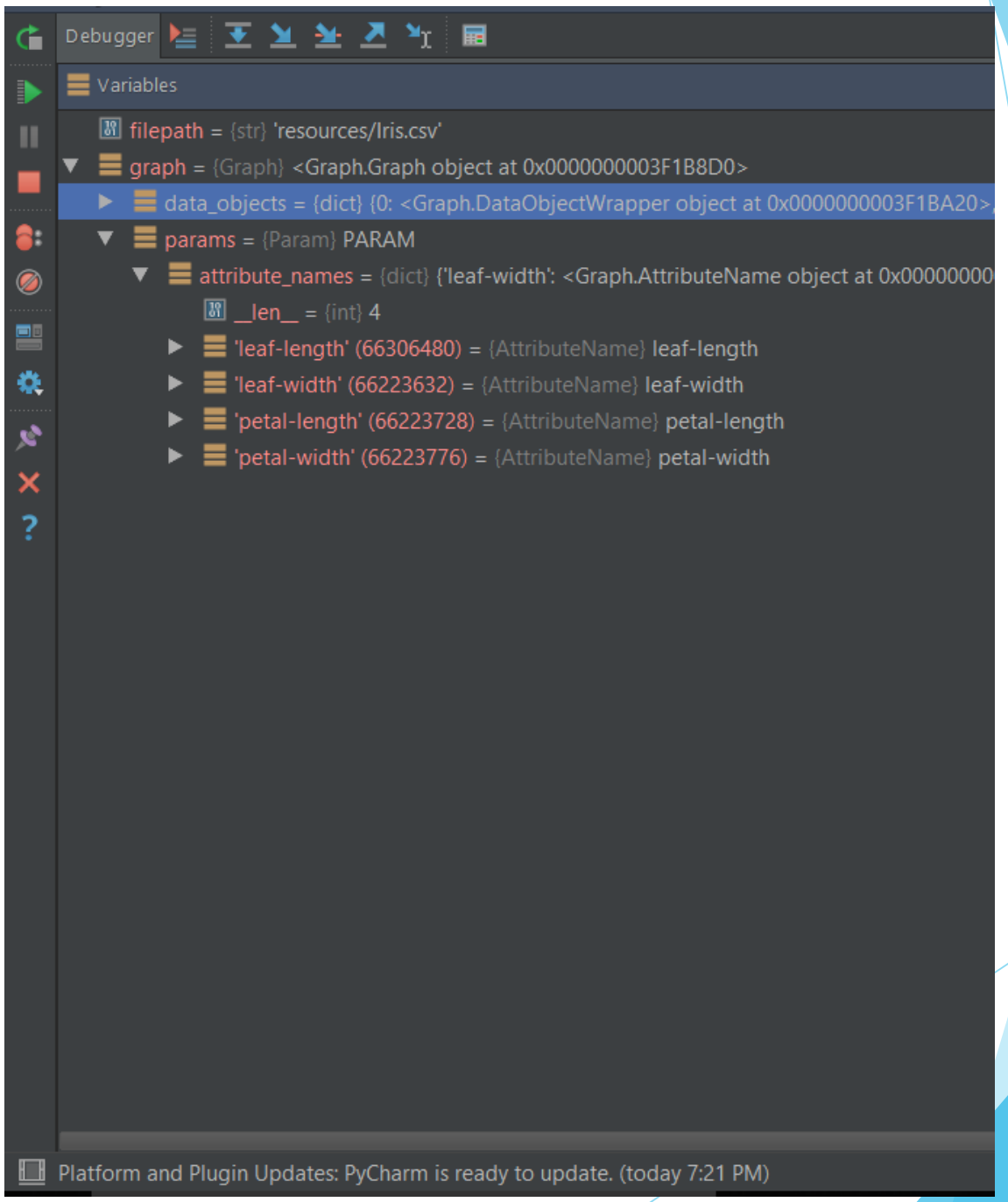
Wartość podobieństwa inkrementowana jest o 1,
w przypadku gdy oba badane obiekty mają tą samą wartość danego parametru

Wartości w tabeli mogą być tylko całkowite z zakresu (0,n),
gdzie n – ilość parametrów

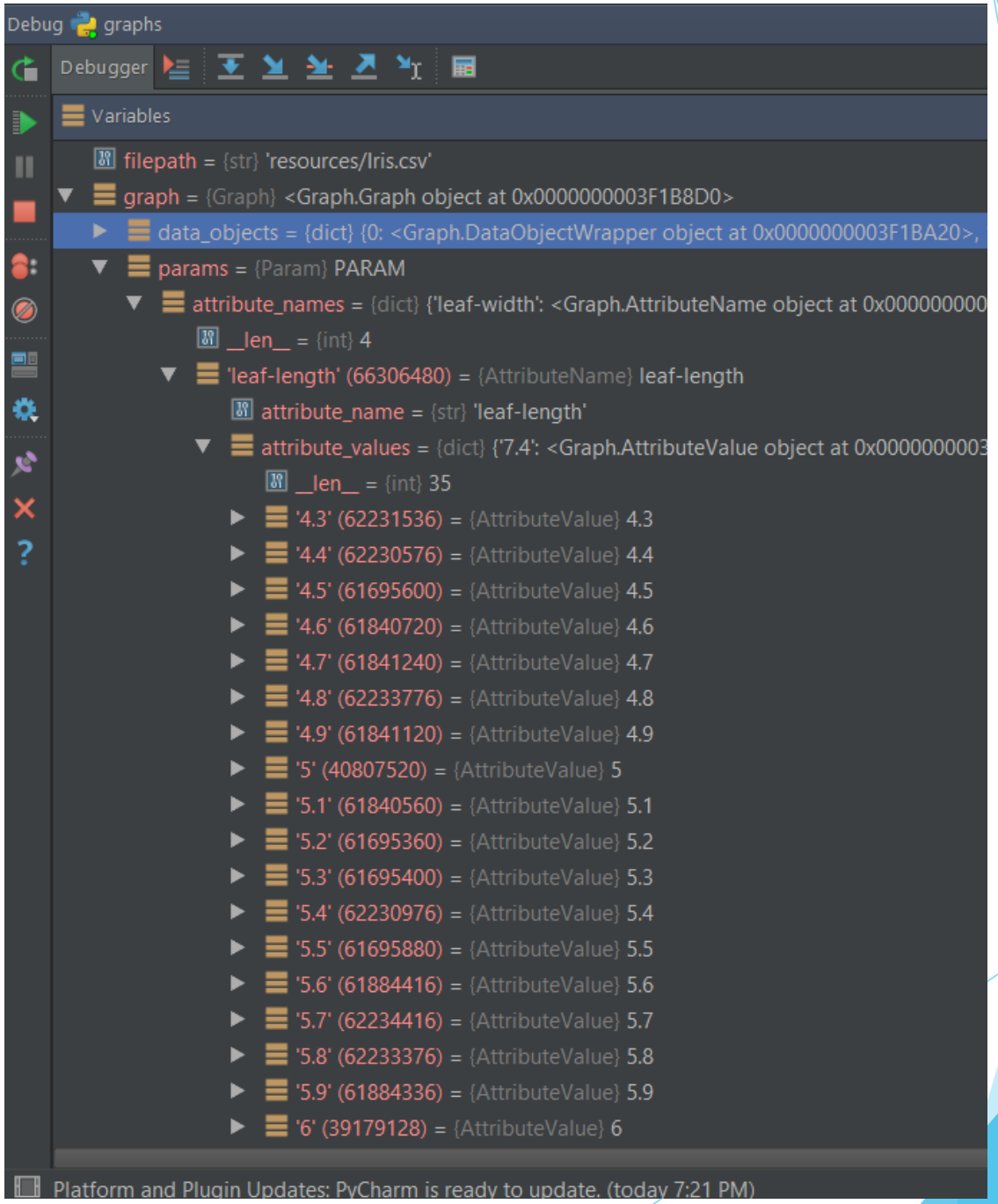
id1

id1	Id2	…	id_n
4 (4 takie same parametry)	3 (3 takie same parametry)		1 (1 taki sam parametr)

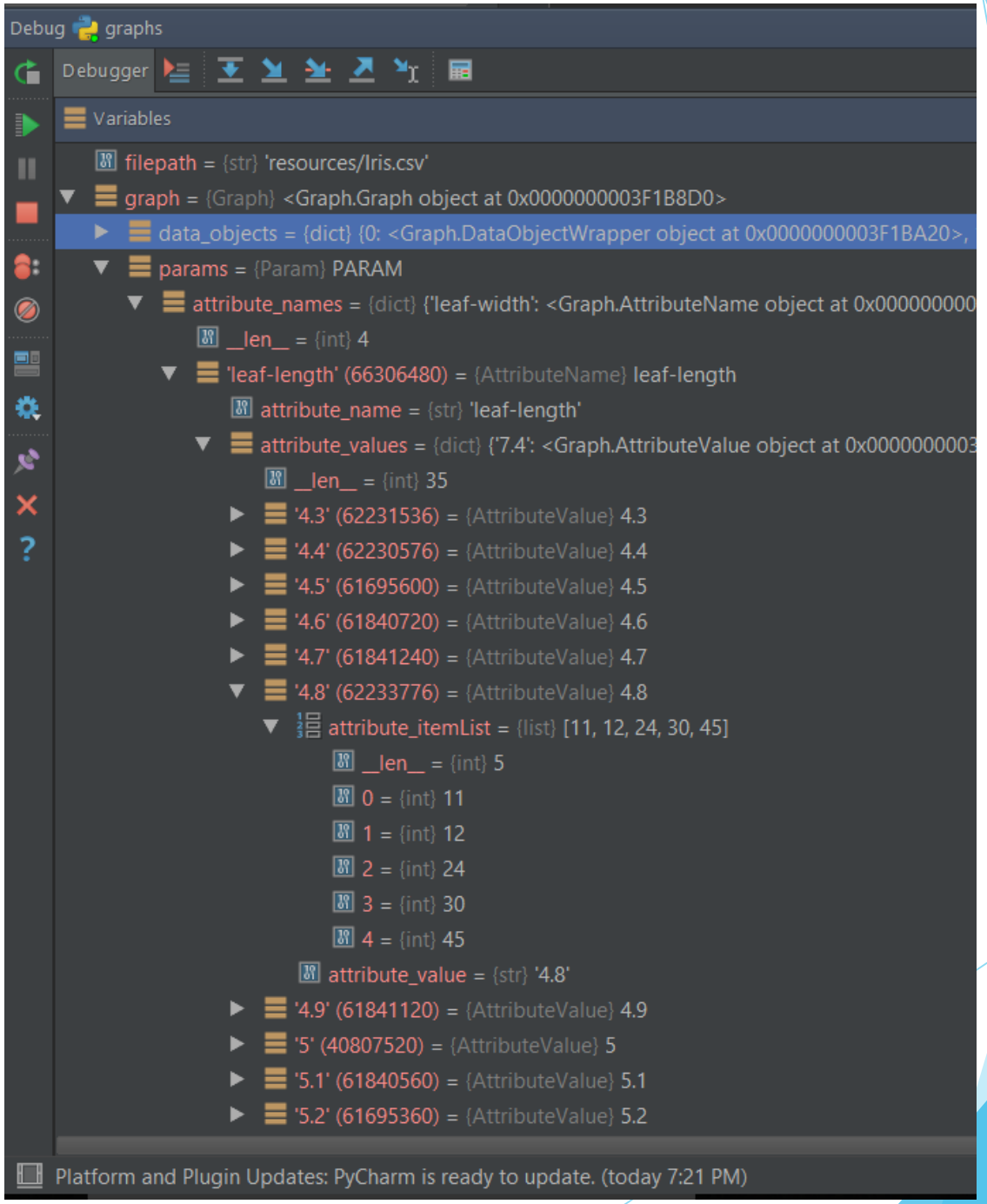
Realizacja


Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>,
params = {Param} PARAM
attribute_names = {dict} {'leaf-width': <Graph.AttributeName object at 0x00000000
__len__ = {int} 4
'leaf-length' (66306480) = {AttributeName} leaf-length
'leaf-width' (66223632) = {AttributeName} leaf-width
'petal-length' (66223728) = {AttributeName} petal-length
'petal-width' (66223776) = {AttributeName} petal-width
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

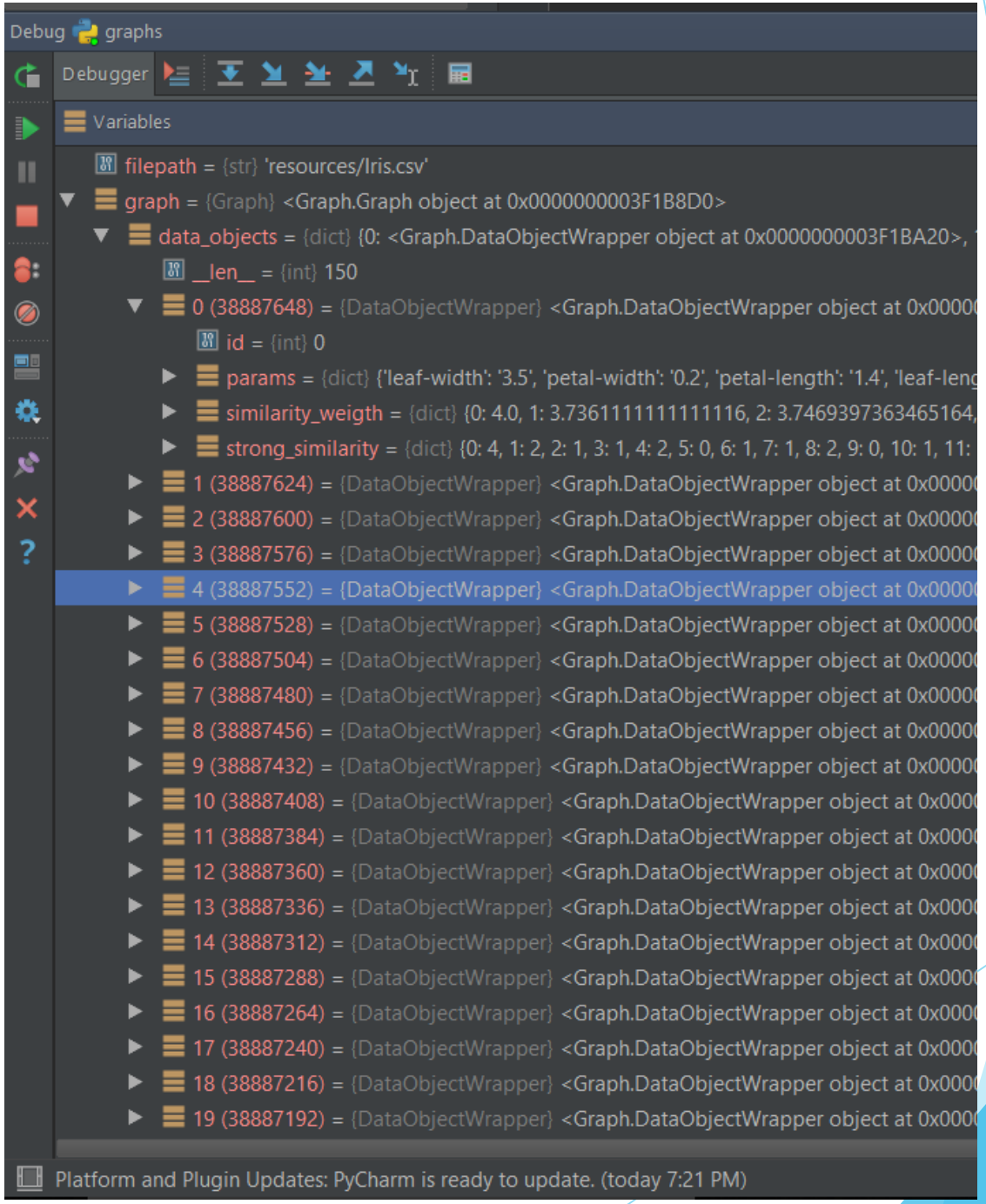
Realizacja


Debug graphs
Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>,
params = {Param} PARAM
attribute_names = {dict} {'leaf-width': <Graph.AttributeName object at 0x00000000
__len__ = {int} 4
'leaf-length' (66306480) = {AttributeName} leaf-length
attribute_name = {str} 'leaf-length'
attribute_values = {dict} {'7.4': <Graph.AttributeValue object at 0x0000000003
__len__ = {int} 35
'4.3' (62231536) = {AttributeValue} 4.3
'4.4' (62230576) = {AttributeValue} 4.4
'4.5' (61695600) = {AttributeValue} 4.5
'4.6' (61840720) = {AttributeValue} 4.6
'4.7' (61841240) = {AttributeValue} 4.7
'4.8' (62233776) = {AttributeValue} 4.8
'4.9' (61841120) = {AttributeValue} 4.9
'5' (40807520) = {AttributeValue} 5
'5.1' (61840560) = {AttributeValue} 5.1
'5.2' (61695360) = {AttributeValue} 5.2
'5.3' (61695400) = {AttributeValue} 5.3
'5.4' (62230976) = {AttributeValue} 5.4
'5.5' (61695880) = {AttributeValue} 5.5
'5.6' (61884416) = {AttributeValue} 5.6
'5.7' (62234416) = {AttributeValue} 5.7
'5.8' (62233376) = {AttributeValue} 5.8
'5.9' (61884336) = {AttributeValue} 5.9
'6' (39179128) = {AttributeValue} 6
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

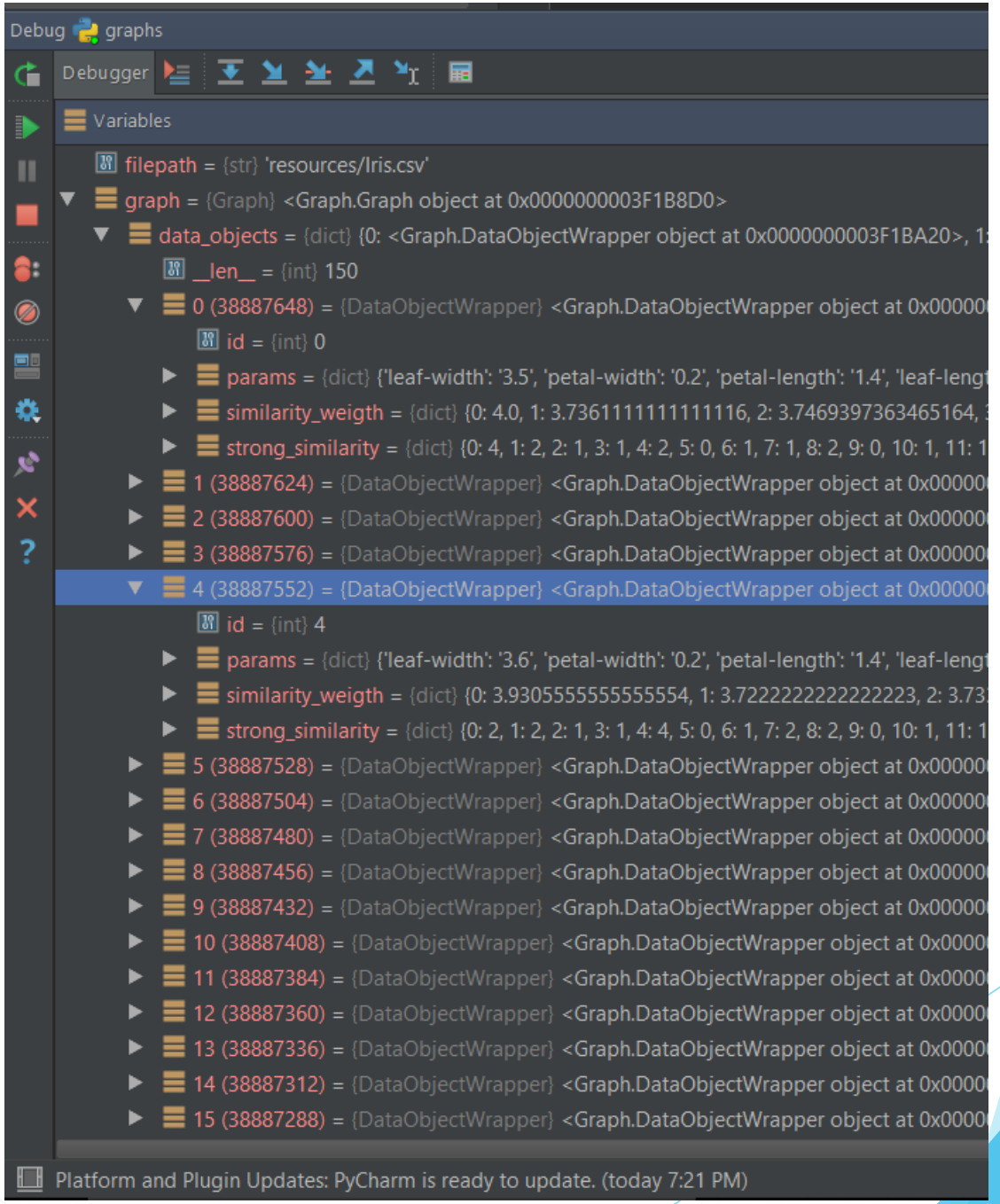
Realizacja


Debug graphs
Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>,
params = {Param} PARAM
attribute_names = {dict} {'leaf-width': <Graph.AttributeName object at 0x00000000
__len__ = {int} 4
'leaf-length' (66306480) = {AttributeName} leaf-length
attribute_name = {str} 'leaf-length'
attribute_values = {dict} {'7.4': <Graph.AttributeValue object at 0x0000000003
__len__ = {int} 35
'4.3' (62231536) = {AttributeValue} 4.3
'4.4' (62230576) = {AttributeValue} 4.4
'4.5' (61695600) = {AttributeValue} 4.5
'4.6' (61840720) = {AttributeValue} 4.6
'4.7' (61841240) = {AttributeValue} 4.7
'4.8' (62233776) = {AttributeValue} 4.8
attribute_itemList = {list} [11, 12, 24, 30, 45]
__len__ = {int} 5
0 = {int} 11
1 = {int} 12
2 = {int} 24
3 = {int} 30
4 = {int} 45
attribute_value = {str} '4.8'
'4.9' (61841120) = {AttributeValue} 4.9
'5' (40807520) = {AttributeValue} 5
'5.1' (61840560) = {AttributeValue} 5.1
'5.2' (61695360) = {AttributeValue} 5.2
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

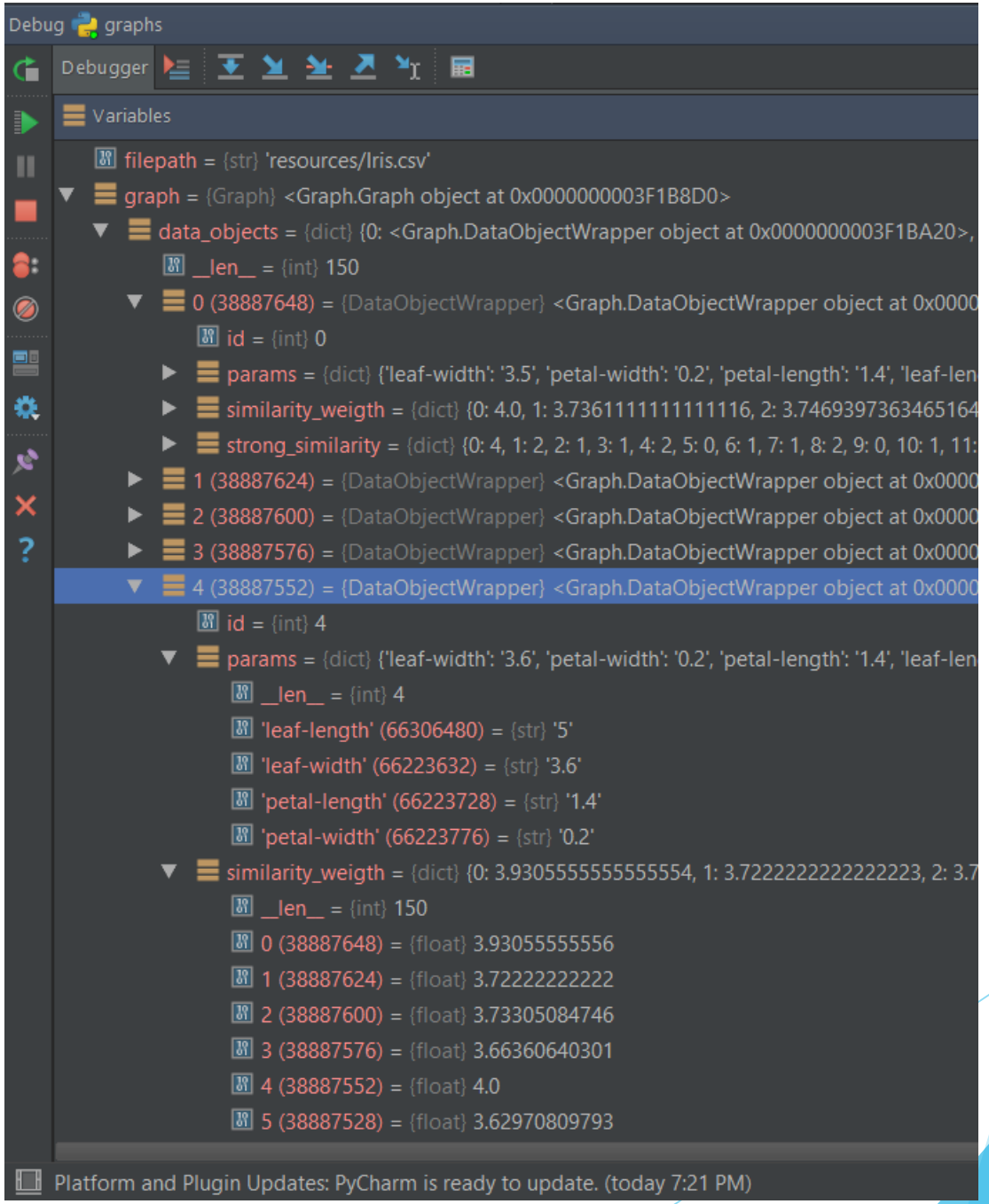
Realizacja


Debug graphs
Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>,
__len__ = {int} 150
0 (38887648) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
id = {int} 0
params = {dict} {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-leng
similarity_weigth = {dict} {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164,
strong_similarity = {dict} {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11:
1 (38887624) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
2 (38887600) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
3 (38887576) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
4 (38887552) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
5 (38887528) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
6 (38887504) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
7 (38887480) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
8 (38887456) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
9 (38887432) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
10 (38887408) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
11 (38887384) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
12 (38887360) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
13 (38887336) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
14 (38887312) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
15 (38887288) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
16 (38887264) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
17 (38887240) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
18 (38887216) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
19 (38887192) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

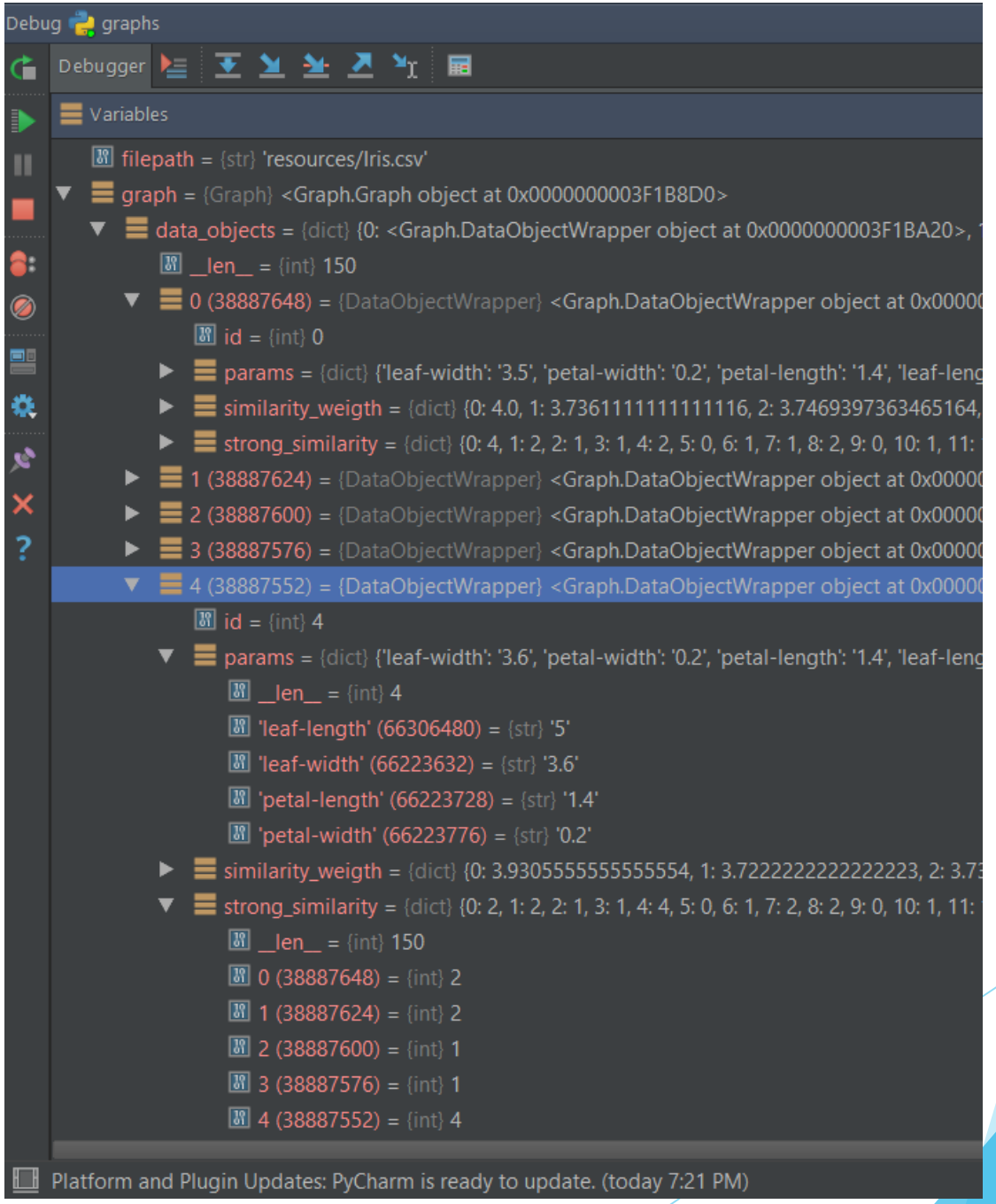
Realizacja


Debug graphs
Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 1:
__len__ = {int} 150
0 (38887648) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
id = {int} 0
params = {dict} {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-lengt
similarity_weigth = {dict} {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164,
strong_similarity = {dict} {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11: 1
1 (38887624) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
2 (38887600) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
3 (38887576) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
4 (38887552) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
id = {int} 4
params = {dict} {'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-lengt
similarity_weigth = {dict} {0: 3.9305555555555554, 1: 3.7222222222222223, 2: 3.73
strong_similarity = {dict} {0: 2, 1: 2, 2: 1, 3: 1, 4: 4, 5: 0, 6: 1, 7: 2, 8: 2, 9: 0, 10: 1, 11: 1
5 (38887528) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
6 (38887504) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
7 (38887480) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
8 (38887456) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
9 (38887432) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x00000
10 (38887408) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
11 (38887384) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
12 (38887360) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
13 (38887336) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
14 (38887312) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
15 (38887288) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

Realizacja


Debug graphs
Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>,
__len__ = {int} 150
0 (38887648) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
id = {int} 0
params = {dict} {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-len
similarity_weigth = {dict} {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164
strong_similarity = {dict} {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11:
1 (38887624) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
2 (38887600) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
3 (38887576) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
4 (38887552) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
id = {int} 4
params = {dict} {'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-len
__len__ = {int} 4
'leaf-length' (66306480) = {str} '5'
'leaf-width' (66223632) = {str} '3.6'
'petal-length' (66223728) = {str} '1.4'
'petal-width' (66223776) = {str} '0.2'
similarity_weigth = {dict} {0: 3.9305555555555554, 1: 3.7222222222222223, 2: 3.7
__len__ = {int} 150
0 (38887648) = {float} 3.93055555556
1 (38887624) = {float} 3.72222222222
2 (38887600) = {float} 3.73305084746
3 (38887576) = {float} 3.66360640301
4 (38887552) = {float} 4.0
5 (38887528) = {float} 3.62970809793
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

Realizacja


Debug graphs
Debugger
Variables
filepath = {str} 'resources/Iris.csv'
graph = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>,
__len__ = {int} 150
0 (38887648) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
id = {int} 0
params = {dict} {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-leng
similarity_weigth = {dict} {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164,
strong_similarity = {dict} {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11:
1 (38887624) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
2 (38887600) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
3 (38887576) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
4 (38887552) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000
id = {int} 4
params = {dict} {'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-leng
__len__ = {int} 4
'leaf-length' (66306480) = {str} '5'
'leaf-width' (66223632) = {str} '3.6'
'petal-length' (66223728) = {str} '1.4'
'petal-width' (66223776) = {str} '0.2'
similarity_weigth = {dict} {0: 3.9305555555555554, 1: 3.7222222222222223, 2: 3.73
strong_similarity = {dict} {0: 2, 1: 2, 2: 1, 3: 1, 4: 4, 5: 0, 6: 1, 7: 2, 8: 2, 9: 0, 10: 1, 11:
__len__ = {int} 150
0 (38887648) = {int} 2
1 (38887624) = {int} 2
2 (38887600) = {int} 1
3 (38887576) = {int} 1
4 (38887552) = {int} 4
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

Funkcjonalności

- Dodanie nowego obiektu do grafu
- Szukanie obiektów o zadanej wartości danego atrybutu
- Filtracja obiektów po wartościach zadanych atrybutów
- Znajdywanie obiektu po id
- Znajdywanie najbardziej podobnych n obiektów do obiektu o zadanym id
- Znajdywanie najbardziej podobnych n obiektów do obiektu o zadanych atrybutach
- Znajdywanie najmniej podobnych n obiektów do obiektu o zadanym id
- Znajdywanie najmniej podobnych n obiektów do obiektu o zadanych atrybutach
- Znajdywanie obiektu o maksymalnej wartości zadanego atrybutu
- Znajdywanie obiektow o wartosciach atrybutu z podanego zakresu

Porównanie czasowe

Funkcja	Bez grafu	Z zastowaniem grafu
Obiekty o zadanym parametrze	0.00169351051901	2.99357920027e-06
Obiekty o zadanych parametrach	0.000679542478461	1.19743168011e-05
10 obiektow najbardziej podobnych do obiektu o zadanym id	0.125938620987493	0.00565144987594
10 obiektow najbardziej podobnych do obiektu o zadanych parametrach	0.634647574562377	0.0151462277908
Obiekt o najwiekszej wartości zadanego parametru	0.009387928783985	2.99357920027e-06
Obiekty o parametrach w zadanym przedziale	0.019794387329487	3.46399878888e-05

Podsumowanie

Zastosowanie struktur AGDS znacznie przyspieszyło pracę algorytmu. Dla niektórych funkcjonalności możemy zaobserwować nawet, że algorytm działa około 1000 razy szybciej!

Ponadto zastosowanie grafu pozwala na przejrzyste przedstawienie danych I ich łatwe przeszukiwanie.

Eksploracja Danych

Eksploracja danych

- Wprowadzenie zbioru danych
- Obliczenie wsparcia dla danych – częstotliwości wystąpień wzorca lub zbioru elementów

```
ZADANIE 1
{'kawa': 0.4444444444444444, 'ser': 0.3333333333333333, 'maslo': 0.3333333333333333, 'cukier': 0.4444444444444444, 'jajka': 0.4444444444444444,
```

- Określenie progu wsparcia, na którego podstawie usunięte są zbiory rzadkie

```
{'kawa': 0.4444444444444444, 'ser': 0.3333333333333333, 'maslo': 0.3333333333333333, 'cukier': 0.4444444444444444, 'jajka': 0.4444444444444444,
```

- Określenie reguł asocjacyjnych dla zbiorów częstych

```
{('chleb', 'maslo'): (0.4444444444444444, 0.6666666666666666), ('chleb', 'miod'): (0.5555555555555556, 0), ('orzeszki', 'jajka'): (0.6666666666666667, 0.3333333333333333),
```

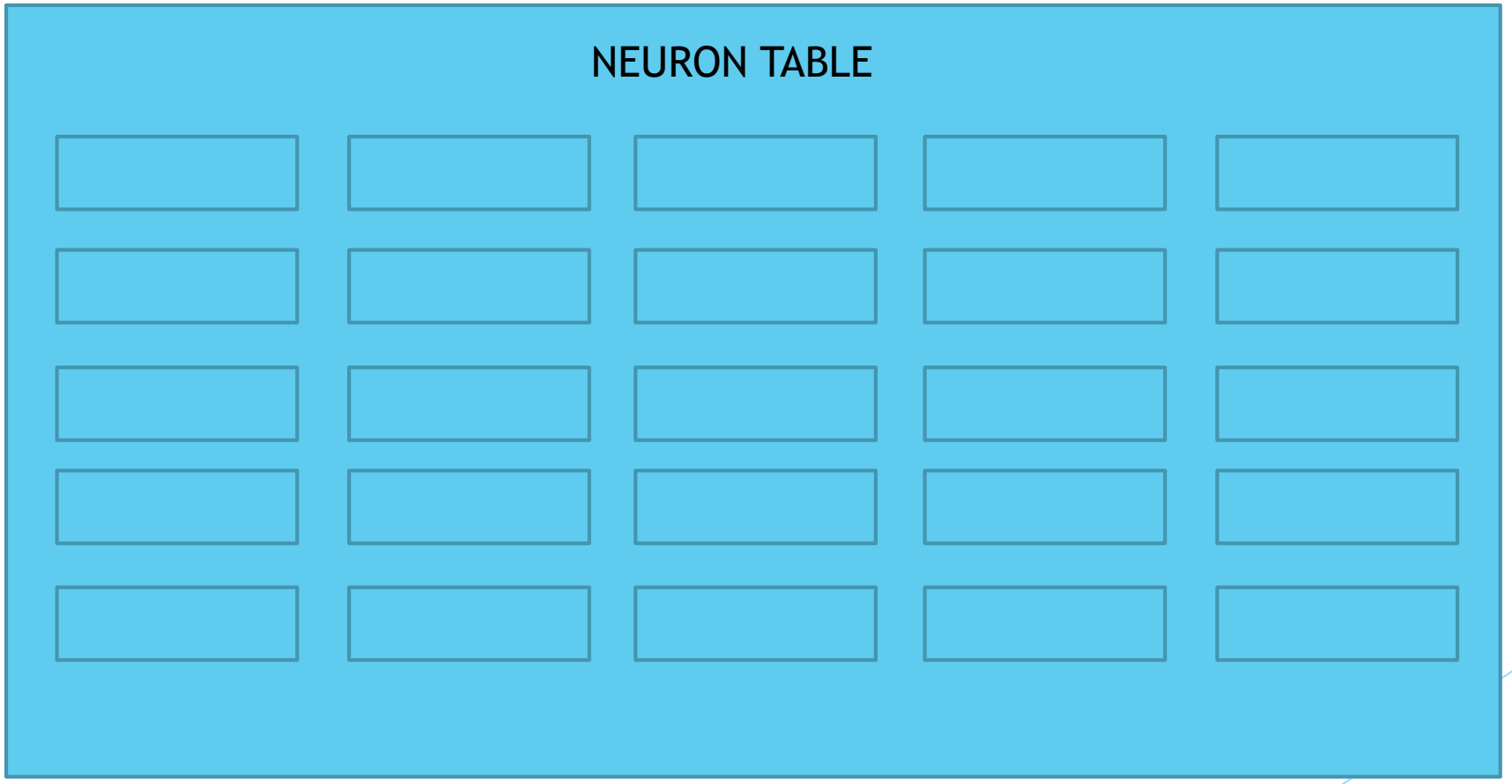
- Ekwiwlentna transformacja klas

```
{'kawa': [1, 3, 2, 6], 'ser': [3, 8, 9], 'maslo': [5, 8, 3], 'cukier': [2, 1, 4, 3], 'jajka': [4, 5, 2, 8], 'miod': [4, 7], 'platki': [4, 7],
```

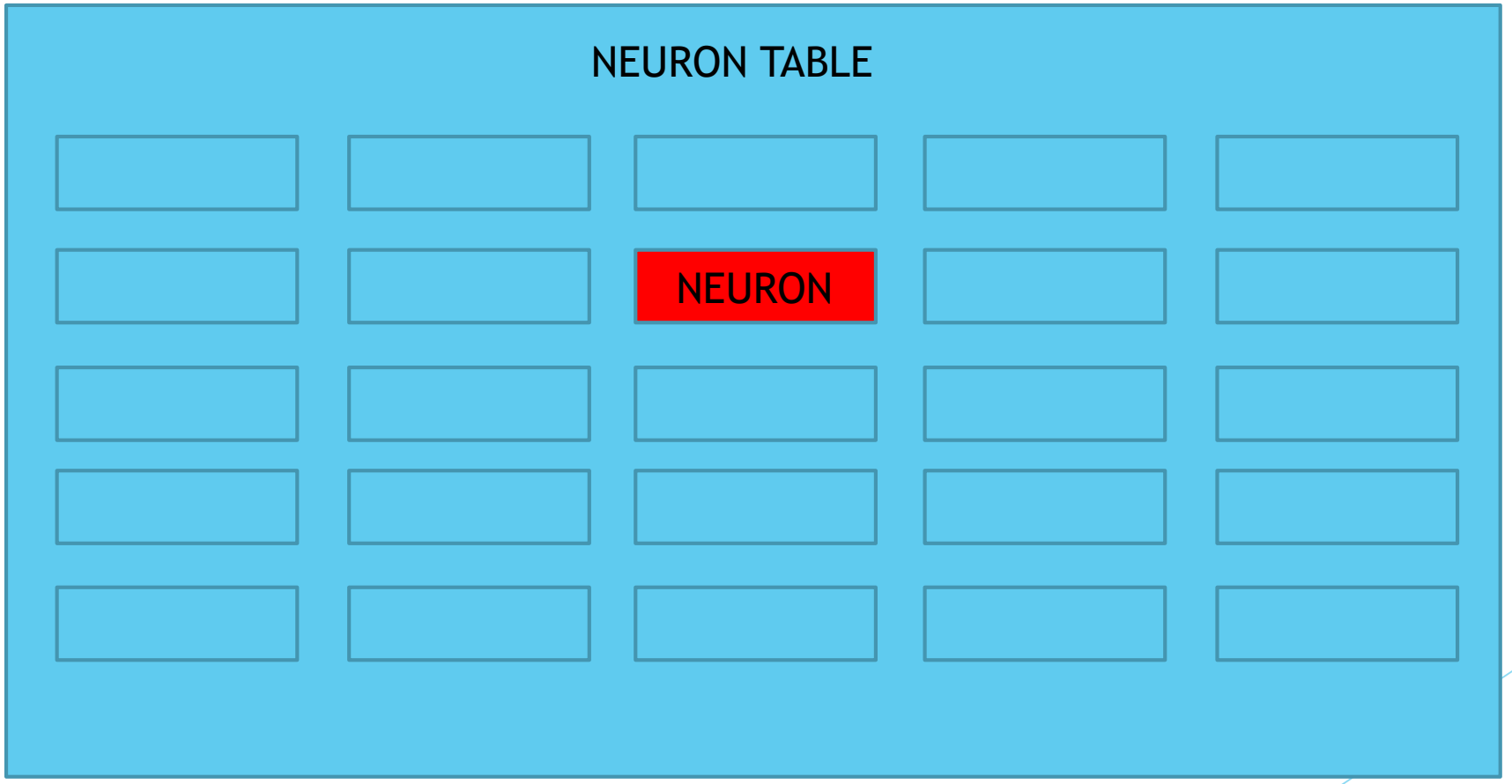
SOM-MAPY SAMOORGANIZUJĄCE SIĘ

SOM to przykład sieci neuronowych uczony metodą uczenia nienadzorowanego, czyli bez określenia np. klasy, do której należą wzorce wejściowe. Sieć SOM na podstawie podobieństwa wzorców wejściowych aktualizuje wagi neuronów na zasadach rywalizacji w taki sposób, iż neuron najbardziej pobudzony ze względu na największe podobieństwo do wzorca uczącego staje się zwycięzcą, którego wagi zostają najmocniej zmodyfikowane w porównaniu z sąsiednimi neuronami, dla których przyjmujemy zwykle mniejsze modyfikacje wag. Dzięki modyfikacjom wag sąsiednich neuronów uzyskujemy możliwość sąsiedniej reprezentacji podobnych klas wzorców.

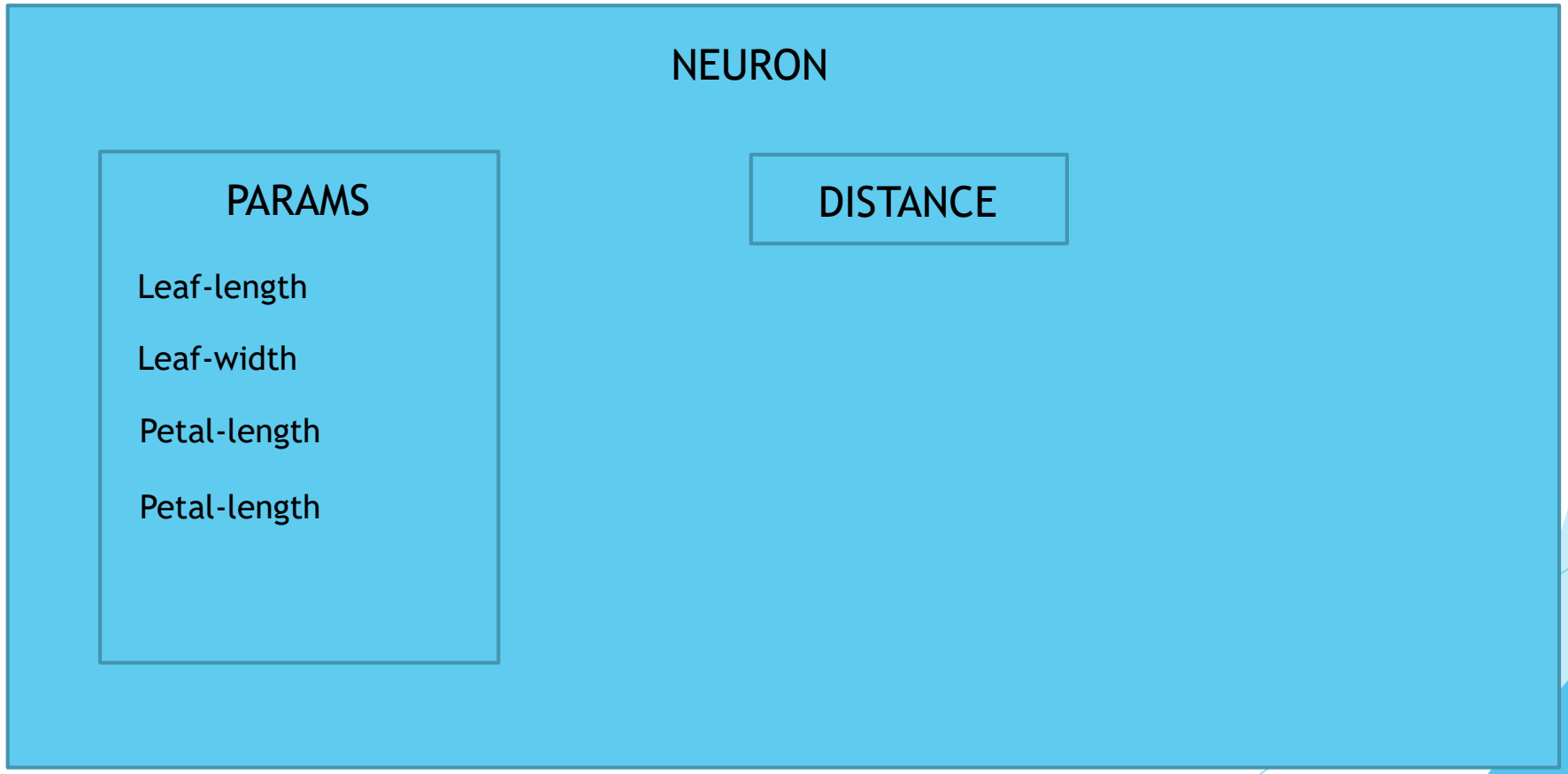
SOM-MAPY SAMOORGANIZUJĄCE SIĘ



SOM-MAPY SAMOORGANIZUJĄCE SIĘ



SOM-MAPY SAMOORGANIZUJĄCE SIĘ



SOM-MAPY SAMOORGANIZUJĄCE SIĘ

1. Utworzenie tablicy neuronów z parametrami zainicjalizowanymi randomowo
2. Obliczenie odległosci kazdego elementu ze zbioru uczącego do każdego z neuronów w tablicy
3. Wybranie neuronu najbliższego dla neuronu testowanego
4. Aktualizacja parametrów w neuronach. Najbardziej aktualizowany jest neuron najbliższy. Parametry sąsiadów neurona najmocniej pobudzonego zmieniane są zgodnie ze wzorem biorącym pod uwagę wage wyliczoną z odległosci danego sasiada od neurona najbardziej pobudzonego

Walidacja krzyżowa

Walidacja krzyżowa

Walidacja krzyzowa to metoda polegajaca na podziale zbioru na zbior uczacy i walidacyjny

Kazdy z elementów zbioru uczącego (niebieski), ma zaimplementowaną tablicę:

V1	
V2	
V3	

Suma odległosci badanego elementu od poszczegolnych elementów ze zbioru walidacyjnego, o klasie V1, podzielona przez liczbę zsumowanych elementów

Walidacja krzyżowa

- Dzięki takiej implenetacji, możemy zbadać do jakiej kalsy należą element zbioru uczącego.
- W moim przypadku po przelosowaniu zbiorow walidacyjnych i uczacych dla k = 10 otrzymywana skuteczność wynosci 95-100%.
- dla k = 20 skutecznosc wynosila od 90-100%.

- Przy zwiększającym się k skuteczność spada ponieważ zmniejsza się liczba elementów zbioru walidacyjnego
- Ostatnim etapem było obliczenie optymalnego k, dla którego skuteczność działania walidacji jest najlepsza. Walidacje przeprowadzono 10 razy na przelosowannych danych. Optymalne k = 8. Jest to największe k o stuprocentowej skuteczności.

KNN

- Podstawowa metoda KNN polega na podziale zbioru na zbiór uczący I walidacyjny:

k

- Następnie obliczana jest odległosc danego elementu ze zbioru uczącego do elementu ze zbioru walidacyjnego. W kolejnym kroku obliczana jest ilość wystąpień poszczegołnych klas pośród k najblizszych elementow. Klasa która najczęsciej się powtarza przypisywana jest elementowi badanemu.
- Skuteczność tej metody jest znacznie mniejsza niz walidacji krzyżowej. Otrzymywane przez mnie wartości skuteczności są rzędu 70-90%.

Dziękuję za uwagę