

# SAS Hash Tables w pigułce

## Spis treści

1.	Wstęp.....	2
2.	Składnia Hash tables.....	2
2.1.	Deklaracja obiektu hash table .....	2
2.2.	Metody w hash table.....	3
2.3.	Atrybuty hash table .....	4
3.	Iteracyjny dostęp do danych .....	4
3.1.	Deklaracja obiektu hiter .....	4
3.2.	Metody obiektu hiter .....	4
4.	Przykłady.....	5
4.1.	Deklarowanie hash table .....	5
4.2.	Użycie iteratora .....	5
4.3.	Ładowanie i szukanie.....	6
4.4.	Sortowanie danych wynikowych .....	7

## 1. Wstęp

Hash Tables znajduje swoje zastosowanie wszędzie tam, gdzie istnieje potrzebna przyspieszenia wykonywania operacji oraz zwiększenia wydajności przetwarzania danych. Pozwala w znaczący sposób przyspieszyć takie operacje jak łączenie tabel, wyszukiwanie podzbiorów oraz agregacje danych. Dzieje się tak, ponieważ technika ta kładzie nacisk na optymalne wykorzystanie pamięci poprzez tworzenie dedykowanej struktury kluczy przechowywanej w pamięci, co pozwala na szybki odczyt wartości kluczy.

## 2. Składnia Hash tables

### 2.1. Deklaracja obiektu hash table

Hash tables można zadeklarować wykorzystując jeden z dwóch sposobów. Stosując wyrażenie `declare`, a następnie operator `_new_`:

```
declare hash obiekt;  
obiekt= _new_ hash();
```

lub używając tylko wyrażenia `declare`:

```
declare hash obiekt();
```

Poniżej znajduje się lista opcjonalnych argumentów, które mogą być wykorzystane w wyrażeniu `declare`:

*dataset*: Określa jaki zbiór ma być załadowany do hash table.

*hashexp*: Określa rozmiar hash table. Liczba 'n' jest potęgą liczby 2. Obiekt hash będzie składał się z  $2^n$  kubełków. Domyślnie  $n=8$ , zatem hash ma 256 kubełków.

*ordered*: Określa w jaki sposób mają być posortowane dane przechowywane wewnątrz hash table:

- "a" | "ascending" | "YES" | "Y" - porządek rosnący
- "d" | "descending" – porządek malejący
- "NO" | "N" – brak porządku (wartość domyślna)

*duplicate*: Określa czy podczas ładowania danych do hash table należy ignorować duplikaty kluczy. Żadna z poniższych opcji nie jest domyślnie ustawiona.

- "replace" | "r" – przechowuje ostatnią wartość duplikatu klucza
- "error" | "e" – zwraca błąd do loga w przypadku gdy zostanie znaleziony duplikat klucza

*multidata*: Określa czy możliwe jest zastosowanie w kluczu wielokrotnych wartości danych.

- "YES" | "Y" – umożliwia wiele wartości dla każdego klucza
- "NO" | "N" – pozwala tylko na jeden element danych dla każdego klucza (wartość domyślna)

## 2.2. Metody w hash table

Metoda `defineKey` pozwala na zadeklarowanie nazw zmiennych, które tworzą klucz:

```
rc=obj.defineKey("klucz1",...,"klucz_n");  
rc=obj.defineKey(all:"yes");
```

Metoda `defineData` pozwala na zadeklarowanie nazw zmiennych, których wartości skojarzone są z kluczem:

```
rc=obj.defineData("zmienna_1",...,"zmienna_n");  
rc=obj.defineData(all:"yes");
```

Metoda `defineDone` kończy fazę deklarowania i inicjalizowania hash table:

```
rc=obj.defineDone();
```

Metoda `add` dodaje do hash table dane i zapisuje je pod daną kombinacją kluczy:

```
rc=obj.add();  
rc=obj.add(key:"klucz1",...,key:"klucz_n"  
           data:"zmienna_1",...,data:"zmienna_m");
```

Metoda `replace` dodaje do hash table dane i zapisuje je pod daną kombinacją kluczy. Jeśli w hash table znajdują się już dane o danym kluczu to zostaną one napisane:

```
rc=obj.replace();  
rc=obj.replace(key:"klucz1",...,key:"klucz_n"  
              data:"zmienna_1",...,data:"zmienna_m");
```

Metoda `find` wyszukuje w hash table wpis odpowiadający podanej kombinacji kluczy i w przypadku, gdy zakończy się sukcesem zwraca zmienną do wektora PDV:

```
rc=obj.find();  
rc=obj.find(key:"klucz1",...,key:"klucz_n");
```

Metoda `check` wyszukuje w hash table wpis odpowiadający podanej kombinacji kluczy i w przypadku, gdy zakończy się sukcesem zwraca wartość 0:

```
rc=obj.check();  
rc=obj.check(key:"klucz1",...,key:"klucz_n");
```

Metoda `remove` usuwa z hash table rekord o podanych wartościach klucza:

```
rc=obj.remove();  
rc=obj.remove(key:"klucz1",...,key:"klucz_n");
```

Metoda `output` wypisuje wartość hash table do zbioru wynikowego:

```
rc=obj.output(dataset:"zbior_wynikowy");
```

Metoda `clear` usuwa wpisy z obiektu hash bez usuwania obiektu hash:

```
rc=obj.clear();
```

Metoda `sum` pobiera sumę dla danego klucza i przetrzymuje go w data stepie pod zmienną `sum_var`:

```
rc=obj.sum();  
rc=obj.sum(key:"klucz1",...,key:"klucz_n"  
          sum:sum_var);
```

Metoda `ref` wykonuje operację wyszukiwania aktualnego klucza. Jeśli klucza nie ma w obiekcie hash to zostanie on dodany:

```
rc=obj.ref();  
rc=obj.ref(key:"klucz1",...,key:"klucz_n");
```

Metoda `equals` określa czy dwa obiekty hash są równe. Jeśli są równe to zmienna `res_var` zostanie ustawiona na 1. W przypadku gdy nie będą równe zmienna ta przyjmie wartość 0 :

```
rc=obj.equals(hash:'hash_obj'. Result: res_var);
```

### 2.3. Atrybuty hash table

Atrybut `num_items` przechowuje informację o ilości elementów trzymanych w hash table:

```
i=obj.num_items;
```

Atrybut `item_size` uzyskuje rozmiar elementu w bajtach:

```
i=obj.num_items;
```

Atrybut `delete` usuwa cały obiekt hash table z pamięci:

```
rc=obj.delete();
```

## 3. Iteracyjny dostęp do danych

### 3.1. Deklaracja obiektu hiter

Iterator pozwala na sekwencyjny dostęp do danych. Aby można było go użyć należy pierw zadeklarować obiekt. Obiekt typu hiter deklarujemy używając poniższej składni:

```
declare hiter nazwa_obiektu_hiter("nazwa_obiektu_hash");
```

### 3.2. Metody obiektu hiter

Metoda `first` zwraca pierwszy rekord z hash table.

```
rc= nazwa_obiektu_hiter.first();
```

Metoda `last` zwraca ostatni rekord z hash table.

```
rc= nazwa_obiektu_hiter.last();
```

Metoda `next` zwraca następny rekord z hash table.

```
rc= nazwa_obiektu_hiter.next();
```

Metoda `prev` zwraca poprzedni rekord z hash table.

```
rc= nazwa_obiektu_hiter.prev();
```

## 4. Przykłady

### 4.1. Deklarowanie hash table

```
data _null_;
set sashelp.class (obs=1);
declare hash ht(dataset:"sashelp.class");
ht.definekey("name");
ht.definedata(all:"yes");
ht.definedone();
name= "Janet";
rc=ht.find();
put rc= name= sex= age= height= weight=;
rc= ht.find(key:"Philip");
put rc= name= sex= age= height= weight=;
rc= ht.find(key:"Nemo");
put rc= name= sex= age= height= weight=;
run;
```

Output:

```
rc=0 Name=Janet Sex=F Age=15 Height=62.5 Weight=112.5
rc=0 Name=Philip Sex=M Age=16 Height=72 Weight=150
rc=160038 Name=Philip Sex=M Age=16 Height=72 Weight=150
```

### 4.2. Użycie iteratora

```
data _null_;
set sashelp.class (obs=1);
declare hash obj(dataset:"sashelp.class", ordered: "Y");
declare hiter iter_obj("obj");
ht.definekey("name");
ht.definedata(all:"yes");
ht.definedone();
rc=iter_obj.first();
do while (rc=0);
  put name= sex= age= height= weight=;
  rc=iter_obj.next();
end;
run;
```

Pętla wypisuje dane do logu. Wykonuje się dopóki metoda `next()` zwróci liczbę różną od 0, czyli aż wypisane zostaną wszystkie rekordy trzymane w hash table.

```
/* Create Input Data Set */
data patients;
length patient_id $ 16 discharge 8;
input patient_id discharge:DATE9.;
datalines;
```

```
Smith-4123 15MAR2004
Hagen-2834 23APR2004
Smith-2437 15JAN2004
Flinn-2940 12FEB2004
;
/* Load and iterate over hash */
data _null_;
length patient_id $ 16
discharge 8;
declare hash ht(dataset:"patients",
ordered:"ascending");
ht.defineKey("patient_id");
ht.defineData("patient_id",
"discharge");
ht.defineDone();
declare hiter iter("ht");
rc = iter.first();
do while (rc=0);
put patient_id discharge:DATE9.;
rc = iter.next();
end;
run;
```

**Output:**

```
Flinn-2940 12FEB2004
Hagen-2834 23APR2004
Smith-2437 15JAN2004
Smith-4123 15MAR2004
```

### 4.3. Ładowanie i szukanie

```
/* Create Input Data Set */
data names;
length first last title $ 16 born died 8;
input first last born died title & $16.;
datalines;
William Blake 1757 1827 Spring
John Keats 1795 1821 To Autumn
Mary Shelley 1797 1851 Frankenstein
;
/* Load and Find */
data _null_;
length first last title $ 16;
length born died 8;
declare hash ht(dataset:"names");
ht.defineKey("first", "last");
ht.defineData("born", "died", "title");
ht.defineDone();
/* Find John Keats */
first = "John";
last = "Keats";
rc = ht.find();
if rc = 0 then
put "Found " first last title $QUOTE.;
else
put "Not Found " first last;
run;
```

**Output:**

```
Found John Keats "To Autumn"
```

#### 4.4. Sortowanie danych wynikowych

```
/* Add to hash and then output */
data _null_;
length patient_id $ 16 discharge 8;
if _N_ = 1 then do;
declare hash ht(ordered:"a");
ht.defineKey("patient_id");
ht.defineData("patient_id",
"discharge");
ht.defineDone();
end;
infile datalines eof=output;
input patient_id discharge:DATE9.;
ht.add();
/*
ht.add() same as:
ht.add(key:patient_id,
data:patient_id,
data:discharge);
*/
return;
output;
ht.output(dataset:"sorted_ids");
datalines;
Smith-4123 15MAR2004
Hagen-2834 23APR2004
Smith-2437 15JAN2004
Flinn-2940 12FEB2004
;
data _null_;
set sorted_ids;
put patient_id discharge:DATE9.;
run;
```

##### Output:

```
Flinn-2940 12FEB2004
Hagen-2834 23APR2004
Smith-2437 15JAN2004
Smith-4123 15MAR2004
```