Hash Table in SAS

Show Reddy Maramreddy - 10Apr2019

Boehringer Ingelheim

Agenda

- What is Hash Table
- Implementation of Hash Table
- Hash Table with Example
- Hash Table Methods and Attributes
- Advantages of Hash Tables

What are Hash Tables

- Hash Tables as Objects
- In SAS also we can accomplish Object Oriented Programming using Hash Tables
  - Objects consist of things that they can do called methods, information about their current state called properties or attributes.
  - Methods and properties are referenced with “dot” notation.

```
abcd.Find()

itemnum = abcd.num_items
```

- Hash Table is a data structure which resides in memory, not on disk
- The memory required for a hash object is allocated at run time
- These objects enable you to quickly and efficiently store, search, and retrieve data based on lookup keys.

What are Hash Tables cont…

- Hash Table are just like the concept of array, hash programming accesses the memory index tables for faster access to data.
- However, unlike arrays, using hash programming does not need sequential integers to access the data
- It can be done via characters or numbers.

Implementation of Hash Table

- Hash Table programming can only be used in a Data Step
- Implementation of Hash table
  - Declaration
  - Define Key
  - Define Data
  - Closing Define

How do we use Hash Tables in SAS (cont..)

Declaration

Syntax : **DECLARE HASH** *object_name*(*argument_tag-n*: *value-n*);

ex: **DECLARE HASH** *hashobj*(dataset: '*ADSL*');

Defining Keys

- The hash object uses lookup keys to store and retrieve data
- On which variable we need to lookup the dataset? we can have a single variable or also multiple variables
- A key is defined by passing the key variable name to the DEFINEKEY method.

ex: adsl.**DefineKey**('subjid');

adsl.**DefineKey**('subjid','center');

How do we use Hash Tables in SAS (cont..)

Defining Data

- Data can consist of any number of character or numeric DATA step variables.
- Data is defined by passing the data variable name to the DEFINEDATA method

  ex: adsl.**DefineData**('trt01pn','trt01p');

Closing Define

- After you have defined all key and data variables, the DEFINEDONE method is called.

  syntax: *object_name*.**DefineDone**();

  ex: adsl.**DefineDone**();

Example

Merging LAB dataset with ADSL to get the treatments in the resultant dataset

```
DATA  labdata;
      IF 0 THEN SET  ads.ADSL;        1
    IF _N_ = 1 THEN DO;
          Declare Hash  abcd(dataset:'ads. ADSL(keep = usubjid trt01pn trt01p)');
          abcd.DefineKey (‘usubjid');
          abcd.DefineData(‘trt01pn',‘trt01p');
          abcd.DefineDone();
    END;
      SET ads.lab;
      IF abcd.Find() = 0;        3
RUN;
```

2

Example (cont..)

1 **IF** 0 **THEN SET** ads.ADSL;

- Load properties of variables in the hash tables
- Use a SET statement like above or
- Use a LENGTH statement to define the variables, their types and lengths
- This SET statement will be read at compile time, information about the variables in the datasets will be extracted, but the statement will never actually execute due to the “IF 0” condition.

Example (cont..)

2

```
IF _N_ = 1 THEN DO;
    DECLARE HASH  abcd(dataset:'ads.ADSL(keep = usubjid trt01pn trt01p)');
        abcd.DefineKey (‘usubjid');
        abcd.DefineData(‘trt01pn',‘trt01p');
        abcd.DefineDone();
 END;
```

- This is where you tell SAS what dataset you want to be loaded into the memory (DECLARE HASH)
- What are the unique key variables you will be using to join (DefineKey)
- and what other variables from the dataset you want to have available for use in memory. (DefineData)

Example (cont..)

3 **IF abcd.Find() = 0;**

- Here the data step is reading observations from the LAB dataset and using the Key variable specified as “usubjid” to join the hash table.
- SAS now attempts to find matches between “usubjid” and whatever variable was specified in the define key statement from the hash table declaration.
- The “IF” condition in the statement will result in an inner join.
- If the key is found the return code is set to zero.
- If the key is not found, the return code is non-zero. i.e a random number allocated by SAS.

Hash Object Methods

- **obj.DefineKey**('key_var1',…,'key_varN');

  Defines key variables

- **obj.DefineData**('data_var1',…,'data_varN');

  Defines data to be stored in the hash object

- **obj.DefineDone**();

  Indicates that key and data definitions are complete.

- **obj.Add**(key:key_val1,…,key:key_valN,data:data_val1,…,data:data_valN);

  Adds the specified data associated with the given key to the hash object.

- **obj.Find**(key:key_val1,…,key:key_valN);

  Determines whether the given key has been stored in the hash object.

Hash Object Methods

- **obj.Replace();**

  obj.replace(key:key_val1,…,key:key_valN,data:data_val1,…,data:data_valN);
  Replaces the data associated with the given key with new data as specified In data_val1…data_valN.

- **obj.Check();**

  obj.check(key:key_val1,…,key:key_valN);
  Checks whether the given key has been stored in the hash object. The data variables are not updated. Return codes are the same as for find.

- **obj.Remove();**

  obj.remove(key:key_val1,…,key:key_valN);

  Removes the data associated with the given key.

Hash Object Methods

- **obj.Clear();**

Removes all entries from a hash object with out deleting the hash object.

- **obj.Output(dataset:'dataset_name');**

Creates dataset dataset_name which will contain the data in the hash object.

- **obj.Sum(sum:sum_var);**

obj.sum(key:key_val1,…,key:key_valN,sum:sum_var);

Gets the key summary for the given key and stores it in the DATA Step variable sum_var. Key summaries are incremented when a key is accessed.

- **rc=obj.Ref();**

rc=obj.ref(key:key_val1,…,key:key_valN);

Performs a find operation for the current key. If the key is not in the hash object, it will be added.

Hash Object Methods

- **obj.Equals(hash:'hash_obj',result:res_var);**

Determines if two hash objects are equal. If they are equal ,res_var is set to 1, otherwise it is set to zero.

- **obj.Delete();**

Deletes the hash object.

Hash Object Attributes

i=**obj.num_items**;

Retrieves the number of elements in the hash object.

sz=**obj.item_size**;

Obtains the item size, in bytes, for an item in the hash object.

Advantages of Hash Table Programming

- **Speed:** Because the lookup occurs in memory, it happens very fast.
- **Composite Keys:** Multiple columns can be used to lookup data.
- **Complex Keys:** Key values can be numeric or character.
- **Control:** Programming hash objects is very low level, giving the developer full control of what goes where.
- When merging datasets using Hash Tables there is no need to sort or indexed the datasets
- Merge Datasets with different number of Key variables or different keys.

Any questions!