Inventory Data Structure

Manual

1. What is Inventory Data Structure?

Inventory data structure is a makeover of a matrix data structure. Often we encounter tasks where we have keywords and their associated properties. Consider like we have 5 items and each item has associated 10 properties, we want to put them in a table and then perform some operations on it.

A matrix is too tedious for that and a hash table is too much. We need a simpler table to capture this data and give us a platform to do the operations. For this reason comes the inventory data structure.

Essentially, it's a table with keywords and properties. It supports the operations like get all keywords, all properties, search, add new keyword etc. The functionalities are explained in detail in the sections ahead.

All this is part of release number 0. The definition is expected to evolve with more meaningful definition and operation with time. If you have suggestions or contributions to make it better and more meaningful, they are always welcome.

2. Design

The motive of inventory table is to hold the data with keywords and properties. So the design is intuitive. Figure below explains the structure of the table.

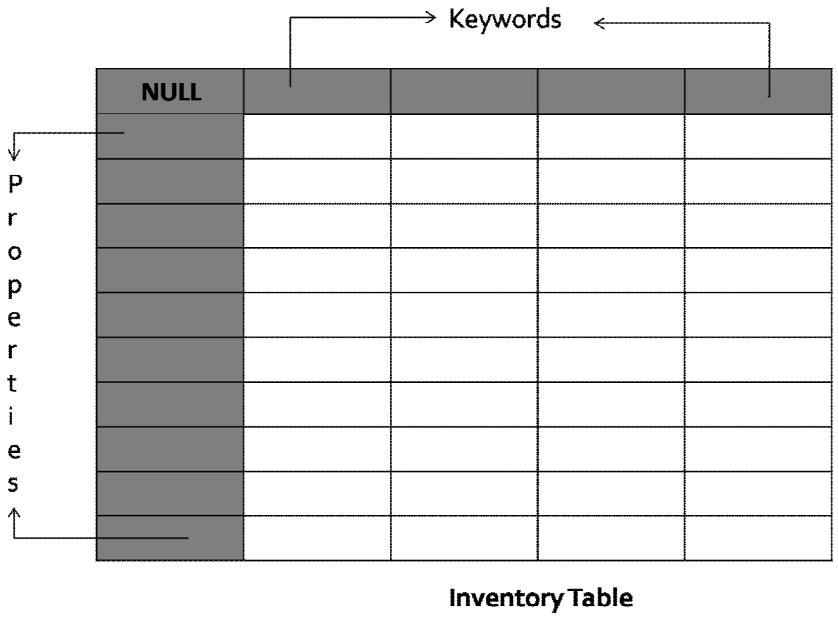

Fig. 1: Layout of the Inventory Table

It's a 3D string array where memory is dynamically allocated based on the number of keywords and properties. The idea is to provide the basic functionalities so that users can later develop the required either by using the existing functions or by building on the table available.

3. Functionality

The design of the table incorporates the following properties:

Function: **add_key**

Description: reads the data from the file and loads the key and the details into the inventory-table present in main memory

Input param: NULL.

Return Type: integer type
success status is returned if key is successfully added to inventory file
failure status otherwise

Function:	**get_keys**
Description:	loads all the keys in the character array
Input param:	NULL
Return Type:	pointer to character array on success character array holding all keys NULL otherwise

Function:	**get_properties**
Description:	loads all the properties in the string array
Input param:	NULL
Return Type:	character type on success character array holding all properties NULL otherwise

Function:	**get_key_properties**
Description:	loads all the properties for the given index of key, from inventory table in the supplied array
Input param:	key's index number for which all its properties has to be loaded
Return Type:	pointer to character array character array is returned holding all properties for given key NULL otherwise

Function:	**get_key_name**
Description:	gets the name of the keyword at the specified location
Input param:	integer value which mentions the index
Return Type:	pointer to character array returns the key if index exists COUNT_EXCEED_ERROR otherwise

Function:	**get_property_name**
Description:	gets the name of the specification at the specified location
Input param:	integer value which mentions the index
Return Type:	pointer to character array
	returns the property if index exists
	COUNT_EXCEED_ERROR otherwise

Function:	**search_key**
Description:	gets the index of the keyword to be searched
Input param:	character array which holds keyword to be searched
Return Type:	pointer to integer array
	returns the keyword index if key exists
	NULL value otherwise

Function:	**search_property**
Description:	gets the index of the property to be searched
Input param:	character array which holds property to be searched
Return Type:	point to integer array
	returns the properties index if property exists
	NULL value otherwise

Function:	**search_any**
Description:	searches for the given word in the entire table, for all the occurrences
	returns indexes of all occurrences of word if search word exists,
	returns -1 otherwise.
	variable to hold indices will be supplied as input param
Input param:	character array, word to be searched and integer array
	to hold the indexes of found locations
Return Type:	NULL

Function:	**print**
Description:	displays all the entries in inventory table
	formatting might not look cleaner if the table size is large
Input param:	NULL
Return Type:	NULL

Below is the prototype of all the functions:

```
int add_key();
char ** get_keys();
char ** get_properties();
char ** get_key_properties(int);
char * get_key_name(int);
char * get_property_name(int);
int * search_key(char array[]);
int * search_property(char property[]);
void search_any(char c_array[], int i_array[100][2]);
void print();
```

The best way to understand is to run the supplied main file and learn the usage.

Note:

It's a PH and VT effort