**Code Flow Graph :**

**IDF\_RANKING :**

* Can define k\_box size in define.
* Calculates idf\_score of tuples having user queried words.
* *Member Variables :*
* *Index index\_obj* -> Index Object to use cell\_index for IDF.
* *map<unsigned int,double> idf\_calc* -> Stores tupleno VS their score in the final k\_box.
* *unsigned int tot\_tups* -> Stores total no of tuples in database for IDF calculation.
* *vector<unsigned int> k\_box ->* Stores tuple no belonging to k\_box.
* *map<string,double> k\_box\_tuples ->* Stores fetched tuples from data file VS their IDF score.
* *string data\_file* -> Stores name of data file to fetch tuple from.
* *Member Methods :*
* *hash\_terms()* :-

-Takes in user query , splits in words and calculates IDF = log(total tuples / freq of that term) for each and every word one by one. Also sends tuple collection and IDF score of each word in query to `calc\_idf`.

* *calc\_idf() :*-

-Fills idf\_calc by finding union of tuplenos in all the words in query and accumulating idf\_score of words into respective tuple no. At the end of this , idf\_calc would have all tuples having any combination of user query words occurring in them, along with tuple idf\_scores.

* *tokenize*() :-

- Sent a vector reference and a line. At the end stores words in vector from line passed splitted by spaces.

* *min\_box\_element() :-*

*-* Returns tupleno with minimum idf\_score in the filled k\_box at a certain point of time.

* *create\_box() :-*

*-* Fills k\_box from idf\_calc with n tuples with highest idf\_scores. Box is filled and then it is checked if coming tuple score is greater than minimum score in k\_box. If no, iteration continues, if yes, the minimum element is replaced with this entry and new minimum of k\_box is found.

* *fetch\_tuples() :*-

- Calls `hash\_terms` and `create\_box`. After obtaining k\_box, it uses tuple\_index of `Index` class to fetch complete tuples from data file. On getting it , tuples along with their respective scores are stored in map - `k\_box\_tuples`.

* *disp\_info :*-

- Displays both `idf\_calc` i.e tupleno Vs IDF\_score and `k\_box\_tuples` i.e tuples VS IDF\_score.

**QFIDF\_RANKING :**

* Calculates qf score of terms in k\_box tuples, based on workload, to break ties of relevancy score as new score = idf + qf.
* *Member Variables :*
* *map<string,double> k\_box\_tuples ->* Stores fetched tuples from data file VS their IDF score.
* *map<double,string> final\_result* ->To store tie-broken tuples with score as key and tuple as value so as to print in descending order of score relevancy.
* *workload windex\_obj* -> workload object so as to use workload column\_cum\_cell index to calculate QF score of each term of tuples in k\_box to break ties.
* *map<int,unsigned int> max\_freq* ->To store RQFmax of all columns in workload’s column\_cell\_index to use for QF = RQF/RQFmax.
* *Member Methods :*
* *start\_qfidf()* :-

-Iterates through each tuple in `k\_box\_tuple` and passes each tuple, one by one, to `tup\_calc()` which gives QF score for passed tuple.This new score is added to old IDF score and then the whole `k\_box\_tuple` is copied to `final\_result` with score as key.

* *calc\_max\_fq() :*-

-Iterates through workload’s column\_cell\_index and fills `max\_freq` with RQFmax for each column. Also can print this RQFmax list.

* *tokenize*() :-

- Sent a vector reference and a line and a delimiter (,). At the end stores words in vector from line passed splitted by the passes delimiter. Used to split tuples of `k\_box\_tuple` into separate columns.

* *tup\_calc() :-*

*-* Receives each tuple of box one by one, breaks each tuple into columns, iterates through them and sends each column to `index\_n\_score`, receives QF score for each column and accumulates them as cumulative score for each tuple to be sent back.

* *index\_n\_score() :-*

*-* Receives each column value and column no.,uses column no. to index from workload’s column\_cell\_index and find RQF of column value that was passed. If column value has multiple words separated by spaces , they are broken into simpler words before indexing.

* *disp\_results() :*-

- Displays `k\_box\_tuple` which has tuple VS (qf+idf) score.Also it displays the `final\_result` having the tie-broken tuples to be presented to users as the result.The score is the key, tuple as value and since map automatically arranges in ascending order of key,function uses a reverse iterator to print the result in reverse, having most relevant result at the top.

***INDEX\_WORKLOAD :***

Defualt Constructor and a Parameterized constructor of workload are defined here.

The Parameterized Constructor takes the file name of the workload and the Index object instantiated. The index object will have the column index, tuple index, cell index and col\_cell index already created.

**MEMBER VARIABLES:**-

* outer\_map Workload\_col\_index:

typedef map<string,vector<unsigned int>> inner\_map;

typedef map<int,inner\_map> outer\_map;

Stores column number as key and stores the strings and query id’s in inner map. A String can have multiple query Id’s.

**MEMBER FUNCTIONS:**

* get\_data\_file()

Return type: Void

Basic Functionality: Opens the workload file passed and delimits each query on “,” using the getline function (Also removes double quotes for each element as it is a necessity because of the format of our CSV File ) to get that specific element and passes each element to identify\_column\_insert\_map function along with the workload\_tid which increments by one for each each query.

As the function name identify\_column\_insert\_map suggests, it identifies the column the element belongs to. If the element is null, it is ignored.

* Identify\_column\_insert\_map
  + Return type: void
  + Input: Inputs the element and user query ID.(I.e., workload\_tid)
  + Basic Functionality: Splits the element one by one if it has spaces, and finds out where each and every part of the full element belongs to which column( i.e.., The column which has the highest frequency of this sub-element) and puts this into a map by using the insert\_in\_map function. For example,if “Highlander 4w 2c” is the element, it breaks it up into three parts i.e.., Highlander, 4w and 2c and finds out individually which column this sub-element belongs to and adds the info into a map using insert\_in\_map function.
* Insert\_in\_map:

Inserts into a map of the type outer\_map the column no, String(Sub element), and tid(query id). Workload col-cell index is being created.

* Display\_workload\_column\_index:
  + Displays the Workload col-cell-index using the outer\_map object.

**INDEX\_DATA**

* **Basic Functionality**
  + The file instantiates an object of Index class. It includes the file “header.h”.
  + The Index class includes index variables which contain the mapping of the data in the database according to various methods of indexing.
  + The class provides indexing methods to create the above mentioned indexes.
* **Class Members**
  + Map objects -
    - Col\_index - a map where, key is column numbers(int) , value is another map(inner\_map). Inner\_map - a map where, key is the values of that column(whole column is taken as a value, delimited only by a comma no delimiting on space character.)(strings) , value is a vector of tuple IDs(vector of ints) corresponding to the tuples where the particular value appears.
    - Col\_cell\_index - similar to col\_index, however the key in the second map(inner\_map) is each individual word in that column, delimiting using space character for every column value(i.e. If a column contains only one value eg.nissan, then nissan is the key. If the column is made up of more than one values, eg. highlander 4WD 4C, then highlander, 4WD and 4C are all separate keys.)
    - Cell\_index - a map where key is every word in the database(every value of every column delimited on both spaces and comma) and value is a vector of tuples denoting where that key appeared.
    - Tup\_index - a map where key is the tuple ID, value is a pair of unsigned int and int, First value of pair is the offset(position in the file where the tuple begins), second value is the length(length of that tuple’s characters.)
    - Total\_size - total number of tuples in the database.
* **Function Definitions**

1. Parameterized Constructor
   1. Input - filename of the database csv file
   2. Retrieves tuples as a single line from the database and passes to the methods create\_column\_index, create\_cell\_index, create\_column\_cell\_index and create\_tuple\_index. While loop runs until there are any more tuples in the database.
2. Copy Assignment
3. Create\_tuple\_index
   1. Input - offset(position of first character of tuple in file), length(length of the tuple in the file), tid(an identifier given to the tuple).
   2. Output - creates the tuple index.
4. Create\_cell\_index
   1. Input - line(the entire tuple, i.e. row of the database file) , TID
   2. Output - creates cell\_index.
5. Create\_column\_index
   1. Input - line(entire tuple), TID
   2. Output - creates column index.
6. Create\_column\_cell\_index
   1. Input - line , TID
   2. Output - creates col\_cell\_index
7. Tokenize
   1. Input - vector of strings v - contains the delimited output of the string(line) passed as argument to tokenize. String delim - contains the characters to be used for the delimiting.
   2. Description - final is a temporary string. The given string is traversed char by char. If the chat is not found in delim, it is appended to the string final. When there are no more chars in the string, in the else part, final is added to the vector of delimited strings. If two consecutive comma chars are found, the string ‘\_E\_M\_P\_T\_Y\_’ is appended to vector v.
8. Insert\_in\_map - used by function create\_col\_index
   1. Input - column no., the word to be inserted in the column, TID, the index in which the word has to be inserted.
   2. Description - the function insert\_in\_map is called every time for each word taken out from the column index. Either a new column is created, a new word is added to an existing column or another entry is made for an already existing word in a column.
9. Display\_column\_index
   1. Prints the column index to a file specified in the function definition.
10. Display\_cell\_index
11. Display\_tuple\_index - similarly