Team 44: database API documentation

Alex Kaiser, Suqian Wang, Dario Avendano

Record Class

The Record class has two attributes, one is a vector of strings called ‘vec_record_’ which saves an individual record, the other is an integer called ‘record_size_’ which is the number of entries of an individual record.

Constructors:

default constructor:

This constructor can be called with no arguments. It is called during initialization which creates an empty string vector.

constructor:

This constructor takes in an argument that specifies the size of an individual record and create a record of the given size with all entries equal to null string.

Destructor:

The destructor is called when the lifetime of an object ends. It will free the resources that the object may have acquired during its lifetime. It might be change depends on whether there is any dynamically allocated memory or pointers in the class.

get_record_size:

This function will return the size of an individual record.

get_record_entry:

This function takes in a given index of the record vector and return the string saved at that index. This function should check if the index is out of range.

set_record_entry:

This function takes in a given index of the record and a string that will store in that entry. It should work for any index, if the index is out of range, increase the vector size. Any empty entry should be initialized to a null string.

Table Class

The Table class has two attributes, one is a vector of strings called ‘columns’ which stores all attributes name (one for each column), the other is a vector of Records called ‘rows’ which stores all individual records (one for each row)

Constructors:

default constructor:

The table class will be able to take in any type of argument of a variable amount or no argument at all. There is nothing returned from the constructor. It creates a blank table.

constructor:

This constructor takes a list of attributes name, passed in as a vector of strings. It will create a table with all pre-defined columns information and no record stored.

add_attribute:

This function will append a new string attribute to the end of the table as a new column. Any entry of that column should be initialized to a NULL value. Nothing is returned as it is a void function. The function should check if the attribute name has already existed, if it has, throws an exception.

delete_attribute:

This function will pass in a string to determine which attribute will be deleted. The function will delete the entire column specified by the given attribute name. Nothing returns from this function as it is a void function. The function should check if the attribute name exists, if not, throw an exception.

insert_record:

The function will check if the key of that record already exists, if it has, finds it in the table and update the record information, if not, append that record to the next empty row of the table. The function will not return a value as it is a void function.

get_attributes-

This function will return all the attributes of a specified table (the private attribute columns stores the information). The return type is a vector of strings. Before return, the function should sort the attributes in order (maybe in lexicographical order). Also, this function will check if the table is empty.

get_size:

This function will return the number of the record in the table. This will work by simply return the size of the private attribute ‘rows’. The function return type is an integer.

return_record:

This function will return the individual record (a row information at some index). This function will check if that record exists, if not, throw an exception. This function return type is Record.

set_key:

This function will give the table a designated string to identify the table. This key will be stored either at the head of the container or in its own separate container. The function does not

return a value as it is a void function. This function will check if the key attribute exists, if not, throw an exception.

count:

This function will count the number of non null entries of a specified attribute. This function should check if the attribute exists. This function returns an integer.

max:

This function will find the maximum value for the given column which is specified by an attribute name. The function should check if the attribute exists. The function returns a string.

min:

This function will find the minimum value for the given column which is specified by an attribute name. The function should check if the attribute exists. The function returns a string.

cross_join:

This function will take in 2 tables as input and return one table as output. The attributes and records of the two tables will be combined into one table and pushed onto the table container.

natural_join:

This function will take in 2 tables as input and return one table as output. The tables must have the same key or else an error will be thrown. The new table will contain a new entry for each row of the first table and columns from the second table.

Database Class

The user is able to instantiate a single database object.

Constructor:

Function without arguments that creates an empty database.

add_table:

It takes a single table object and a string containing the name that the table will be assigned to in the database then it adds it to the database. It is a void function so it doesn't return anything.

drop_table:

Takes in a string that contains the name of a table in the database and deletes it from the database. It is a void function so it doesn't return anything.

list_table:

This function takes in a Database object and returns a vector of strings containing all the names in the database.

get_table:

This function takes in a database object as input and returns a vector of table objects containing all the tables inside the database.

Query:

This function takes in 3 different strings. The first string will provide a list of the attribute names to keep. The second string will take in the name of the table and the final string will reference the attribute names. The function returns a Table object given the commands inside the Query function.