Veena Ghorakavi
Ruben Valdez
Wyatt Klein
CSCE 315
September 23,2017

API Documentation

Database class:

Database()

- database constructor that creates the empty database
- No inputs are needed for this method and returns no values

Database(const Database& d)

- database copy constructor that initializes the database using a database object
- input for this method is an Object of type Database
- Returns no values

~Database()

- Database destructor
- No inputs are needed and returns no values

void add_table(Table object, std::string name)

- adds the table to the database
- Requires 2 inputs, first is a Table object second is the name of the table in the type of a string
- Returns no values

void drop_table(std::string name)

- drops the table from the database
- input is the name of the table to be dropped from the database as a string
- returns no values

vector<std::string> list_table()

- gets the list of tables from the database
- takes in no inputs
- returns a vector of type string

vector<Table> get_table()

- gets all of the tables that are currently in the database
- takes in no input and returns a vector of type Table

Table query_command(std::string select, std::string from, std::string where)

- Query function that can help with selection and finding of the databases
- returns table with queried clauses

Table Class:

Veena Ghorakavi
Ruben Valdez
Wyatt Klein
CSCE 315
September 23,2017

int _size

- gives size of table

std::deque<Record>records

- keeps track of the records in the table

typedef std::vector<std::pair<std::string, Record>>Table_column

- keeps the records and

names associated with the records. Type def

std::vector<Table_column> Table_vec

- Keeps track of the rows of the tables

Table_column column_list

- the actual columns

std::vector<std::string> keys

- keeps track of keys

typedef std::deque<Record>::const_iterator Table_iterator

- keeps track of the position in thetable

Table_iterator begin() const

- function that returns the beginning of the table
- no input is needed
- returns a Table_iterator

Table_iterator end() const

- function that returns the end of the table
- no input is needed
- returns a Table_iterator

Table()

- table constructor initializes the Table
- Requires no input and has no output

Table(const Table& t)

- table copy constructor that initialized a table using a Table object
- Takes in one input that is of Table type
- Returns no values

Veena Ghorakavi
Ruben Valdez
Wyatt Klein
CSCE 315
September 23,2017

Table(int size)

- initializes a table with a specified size
- Takes in one input of type int which serves at the size of the desired table
- Returns no values

Table(const Table_column& columns_list)

- initializes a table with certain columns
- takes in an input of type Table_column
- returns no values

Table(std::string, std::string, std::string)

- constructor that creates a table with 3 attributes
- takes in 3 inputs of type string each input being the attribute name
- returns no values

void add_attribute(std::string table_attribute)

- adds an attribute to the table
- takes in 1 input of type string which is the name of the attribute
- returns no values

void delete_attribute(std::string table_attribute)

- removes the table attribute given the name of the attribute
- takes in one input of type string
- no values are returned

std::vector<std::string> get_attribute_in_order()

- gets attributes of the table in order
- requires no inputs and returns a vector of string

int get_size()

- gets size of table
- requires no input and returns a value of type int

void specify_key(std::string table_key)

- finds key in the table given the table key
- Takes in 1 input of type string
- Returns no values

Table cross_join(Table a, Table b)

- joins two tables and creates one Table
- Takes in two inputs of type Table

Veena Ghorakavi
Ruben Valdez
Wyatt Klein
CSCE 315
September 23,2017

- Returns a value of type Table

Table natural_join(Table a, Table b)

- joins two tables based on key and returns one Table
- Takes in two inputs of type table
- Returns a value of type Table

int compute_count(std::string table_attribute)

- Computes a count on non-null entries in table on an attribute from the table
- Takes in one input of type string which is the name of the attribute to do the count
- Returns a value of type int

int compute_min(std::string table_attribute)

- finds minimum record on an attribute from the table
- takes in one input of type string which is the name of the attribute to find the min
- Returns a value of type int

int compute_max(std::string table_attribute)

- finds maximum record on an attribute from the table
- takes in one input of type string which is the name of the attribute to find the max
- returns a value of type int

Table add(Record r)

- adds record to table and returns table
- Takes in a input of type Record
- Returns an object of type table

Record Class:

int _size

- the size of the record created

typedef std::tuple<std::string, std::string> Record_tuple

- record is a tuple

std::vector<Record_tuple> Record_vec

- create the vector that keeps track of the records

typedef std::vector<std::tuple<std::string, std::string> >::const_iterator Record_iterator

- keeps track of place within records

Record()

Veena Ghorakavi
Ruben Valdez
Wyatt Klein
CSCE 315
September 23,2017

- record constructor; initializes a record
- Takes in no input and returns no values

Record(int size)

- Creates a record with a specified size
- Takes in an input of type int; this is the size of the record to be created
- Returns no values

Record(const Record& r)

- record copy constructor; initialized a record using a record object
- requires one input of type Record
- returns no values

int size()

- gets size of the record
- takes in no inputs
- returns a value of type int

Record(std::vector<std::tuple<std::string, std::string>> objects)

- creates record based on other records; for use when the tuples are used to initialize record
- Takes in one input of vector, touple pair of (string,string)
- Returns no values

~Record()

- Destructor
- Takes in no input and returns no values

std::string get(unsigned int index)

- get a string at an index from the record
- takes in one input of type int; this is the index of the record
- returns a value of type string which is the value of the record

void set(int index, std::string name)

- used to set a a record given the index and the name of the record to create
- takes in 2 inputs, first is a value of type int, second is a value of type string
- returns no values

void set(std::string, std::string, std::string)

- sets a record given three record names
- takes in 3 inputs of type string
- returns no values

Veena Ghorakavi
Ruben Valdez
Wyatt Klein
CSCE 315
September 23,2017

void set(std::tuple<std::string, std::string>)

- used to set a a record given a tuple
- takes in one input of a tuple pair of (string, string)
- returns no values

Record_iterator begin() const

- find beginning of iterator
- takes in no inputs and returns no values

Record_iterator end() const

- find end of iterator
- takes in no inputs and returns no values

std::string operator [] (unsigned int i) const

- overloads the [] operator to find specific record at a specific position in Record vector given the inded
- takes in one input of type int
- returns a value of type string which is the value of the string at the given index