Team 28 (Michael Gyarmathy*, Jayant Notani, Nishaanth Narayanan, Logan Elliott)
CSCE 315

team28-db Documentation

Database

class used to represent a Database
User only needs to include “database.h” to obtain all the functionality.

Database()

- default constructor of a Database
- creates an “empty” Database

Database(const Database &d)

- copy constructor of a Database, used to duplicate a pre-existing Database

Database(string fileName)

- constructor used to create a Database from a file

Database(const Table &t)

- constructor used to create a Database from a pre-existing Table

int addTable(const Table &t, string name)

- an add table function that takes in a single table and a name, and adds that table to the database and returns 1 upon success
- returns 0 if unsuccessful

int dropTable(string name)

- a drop function that takes a table name and deletes it from the Database and returns 1
- returns 0 if table name does not exist in the Database

int save(string filename)

- a save function that takes a filename and saves the database to that file and returns 1 upon sucess
- returns 0 if save is unsuccessful

int load(string filename)

- a load function that takes in a filename and loads in a database from that file
- any pre-existing Database is deleted
- returns 1 upon success
- returns 0 if load is unsuccessful

int merge(const Database& d)

- a merge function that allows another Database to be merged into the pre-existing Database and returns 1 upon success
- returns 0 if merge is unsuccessful

int merge(string fileName)

- a merge function that allows another Database (from a file) to be merged into the pre-existing Database and returns 1 upon success.
- returns 0 if merge is unsuccessful (ie file does not exist)

int copy(const Database& d)

- a copy function that copies an entire Database onto the current Database and returns 1 upon success
- similar functionality to a copy constructor
- returns 0 if copy is unsuccessful

int copy(string fileName)

- a copy function that copies an entire Database (from a file) onto the current Database and returns 1 upon success.
- returns 0 if copy is unsuccessful (ie file does not exist)

vector<string> listTables()

- a list table function that returns a list of all table names in the Database

vector<Table> getTables()

- a get table function that returns a list of all Tables in the Database

Table queryTable(string columnsToSelect, string fromTable, string whereClause)

- a query function that takes three string arguments: SELECT, FROM, WHERE, all passed in as strings
- SELECT allows either a list of which attribute names to keep, or an indicator (*) to keep all attributes
- FROM allows a single table name
- WHERE references to attribute names, allowing for comparisons (=, !=, >, <, >=, and <=), and IN operator (given the name of a table with only one attribute), an EXISTS operator (given the name of a table with only one attribute), AND/OR/NOT, parentheses (up to three levels), an ALL operator (given the name of a table with only one attribute), and an ANY operator (given the name of a table with only one attribute.
- returns a Table containing the queried data

int deleteRows(string fromTable, string whereClause)

- a delete function that deletes rows from from a table where the WHERE clause is met
- The function returns 1 upon success
- returns 0 if delete is unsuccessful

int updateTable(string tableName, string setClause, string whereClause)

- an update function that updates an entry/entries in a given Table (specified by name), to set them to values that are either constant (integers and floats) or a computed function on attribute values (from that table; +, -, *, and / are supported) as specified by the SET clause, where the WHERE clause is true
- returns 1 upon success and returns 0 if update is unsuccessful

Table

class used to represent a table in a database

Table()

- default constructor of a Table

Table(vector<Attribute> columns)

- constructor used to create a Table with a predefined set of attributes

int addColumn(Attribute a)

- an add function that takes in a single Attribute (with name and type), and adds a column to the end of the

Table with that new attribute

- any entries currently in the table should get NULL for that entry
- returns 1 upon success and returns 0 if unsuccessful

int deleteColumn(Attribute a)

- a delete function that takes in a single Attribute and deletes it from the Table
- Function returns 1 upon success
- returns 0 if Attribute does not exist in the Table

int deleteColumn(string attributeName)

- a delete function that takes in a single attribute name and deletes it from the Table. Returns 1 upon success.
- returns 0 if the attribute name does not exist in the Table

int renameColumn(string oldName, string newName)

- a rename attribute command that takes two names, and replaces the name for the attribute given by the first name with the second name
- returns 1 upon successful rename
- returns 0 if the old attribute name does not exist in the Table

vector<Attribute> getColumns()

- a get attributes function that returns a list of Attributes (name and type) for that Table

int insertRow(string values)

- an insert function that takes a record (string of values) and adds it to the Table. Returns 1 upon success.
- returns 0 if unsuccessful

int getNumberOfRows()

- a function that returns the number of rows a Table contains (size)

Record& rowAt(int index)

- a function used to return individual records from the Table

int setKey(vector<Attribute> attributes)

- a function to specify a key with a list of Attributes.
- Returns 1 upon success.
- returns 0 if a given Attribute does not exist in the Table

int setKey(vector<string> attributes)

- a function to specify a key with a list of attribute names
- Returns 1 upon success
- returns 0 if a given attribute name does not exist in the Table

Table crossJoin(const Table& a, const Table&b)

- a cross join function that takes two Tables as input and returns one Table as output

Table naturalJoin(const Table& a, const Table&b)

- a natural join function that takes two Tables as input and returns one Table as output
- one entry is created for each row of the first Table, with additional columns from the matching key in the second Table

string sum(string column)

- returns the sum of values of a given column of the Table
- return type is always string regardless of the type of column.

string min(string column)

- returns the minimum value of a given column of the Table
- return type is always string regardless of the type of column.

string max(string column)

- returns the maximum value of a given column of the Table
- return type is always string regardless of the type of column.

int count(string column)

- returns the number of non-null entries in a given column of the Table

Record

class used to store an individual record (tuple) as a set of strings

Record(vector<string> entries)

- constructor of a Record which requires a vector of strings representing the entries

string& elementAt(int index)

- a function that allows access to an individual entry in the record (given an index) for modifying and retrieving

Attribute

class used to represent a table attribute with both type and name

enum Type {INTEGER, FLOAT, VARSTRING, DATE, TIME}

- enumeration used to designate the type of the Attribute
- can be:
    - INTEGER
    - FLOAT
    - VARSTRING (string of variable length)
    - DATE (in the format YYYY/MM/DD)
    - TIME (in the format HH:MM:SS, 24-hour time)

Type type

- Attribute Type

string name

- Attribute name

Attribute(Type t, string name)

- constructor of an Attribute

- requires a given Type and attribute name