A. **Overview - Logging in the database:**

The logger in the database will create a log file. The name of the file is based on database start time. For instance, if database was started at 2017-04-30, 3:25 PM and 36 seconds, the database log will be named logfile\_2017\_04\_30\_15\_25\_36.txt. All the logs of one session will be recorded in the log file until the database is shut down. On restarting the database executable, a new log with a new time stamp will be created. The log file will be created in the ‘cli’ folder of the project.

Keeping under consideration that disk I/O is expensive, the two factors that are used to limit the number of disk I/O per database execution session are, the number of transactions recorded by the log buffer and the current size of the log buffer.

1. Transaction counter: The transaction counter in the log buffer is increment after call to the updateLog() method to update the log buffer. On reaching the transaction counter threshold (for this project, the threshold is currently defined as 200 in the cli\_logger.h file), the buffer data is written to the disk.
2. Buffer size: The log buffer size is calculated after writing the log data to the buffer. Data is written to the log buffer on each call made to updateLog() method. The buffer is continually updated if the current size of the buffer after update is less than the threshold for the buffer that is defined in the header file (i.e. cli\_logger.h). On reaching the threshold, the data from the buffer is written to the disk, hence emptying the buffer and updating the log file for the database.

Between transaction counter and the current buffer size, whichever threshold is reached sooner will trigger a write to the disk. The reason for a two-factor tracker for the log is, in case there is a situation where one transaction significantly increases the current size of the buffer then current buffer size will check that increase and write to the disk without making the logger wait until transaction counter reaches the set threshold. Similarly, to check for a situation when current buffer size threshold is not reached even after many updates to the current buffer, the transaction size is tracked. Therefore, on reaching the counter threshold, the buffer will be written to the disk even without reaching the buffer size threshold. This avoids a situation where the buffer has been written to massive number of times but the disk has not been updated because the size threshold of the current buffer has not been reached.

The threshold for current buffer size and transaction counter are defined in the cli\_logger.h and can be easily updated or fine-tuned as needed. Furthermore, the size of the buffer is twice the size of any log message buffer. This will ensure that there is never a buffer overflow when writing the log buffer.

B. **Graph database logging:**

The graph database logging functionality keep tracks of the following actions in the database:

1. Creating graph and adding vertex.
2. Creating edge between vertices in a graph.
3. Creating schema for an edge or a vertex.
4. Creating enum.
5. Creating tuple for an edge or a vertex.

1. **Creating graph and adding vertex:**

On creating a graph in the database, the logger will log the time of graph creation, the vertex id that was created in the graph and the graph id of each of the graphs created as follows:

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> g n

grdb> g n

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_15\_48\_52.txt

VERBOSE ->2017-04-30 15:48:52 :: Database Initialization!

VERBOSE ->2017-04-30 15:48:55 :: Adding graph with id 0.

VERBOSE ->2017-04-30 15:48:55 :: In graph 0, adding vertex with id 1.

VERBOSE ->2017-04-30 15:48:56 :: Adding graph with id 1.

VERBOSE ->2017-04-30 15:48:56 :: In graph 1, adding vertex with id 1.

VERBOSE ->2017-04-30 15:48:58 :: Adding graph with id 2.

VERBOSE ->2017-04-30 15:48:58 :: In graph 2, adding vertex with id 1.

VERBOSE ->2017-04-30 15:48:58 :: Shutting down database!

On entering ‘q’ to quit the running instance of the graph database, the logger will log ‘Shutting down database!’ hence keeping track of when the program started and ended.

2. **Creating edge between vertices in the graph:**

The log keeps track of what vertices are being added to which graph. It keeps track of the vertex id being created, graph id and time of creation. The database is set up in a way where if one of the vertices is absent to create an edge, the missing vertex will be added and followed by addition of the edge. The database logger will keep track of it. Following is an example of workings of the graph when adding an edge between existing vertices where one vertex is present and another one is missing. Similarly, the following example also shows the log when both the vertices for an edge are missing.

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> g e 1 2

grdb> g

>0:({1,2},{(1,2)})

grdb> g e 1 3

grdb> g e 3 2

grdb> g e 5 2

grdb> g

>0:({1,2,3,5},{(1,2),(1,3),(3,2),(5,2)})

grdb> g e 8 9

At least one vertex must exist in graph

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_18\_02\_41.txt

VERBOSE ->2017-04-30 18:02:41 :: Database Initialization!

VERBOSE ->2017-04-30 18:02:46 :: Adding graph with id 0.

VERBOSE ->2017-04-30 18:02:46 :: In graph 0, adding vertex with id 1.

VERBOSE ->2017-04-30 18:02:48 :: Found vertex 1 in graph 0.

  ERROR ->2017-04-30 18:02:48 :: Failed to find vertex 2 in graph 0.

VERBOSE ->2017-04-30 18:02:48 :: In graph 0, adding vertex with id 2.

VERBOSE ->2017-04-30 18:02:48 :: Adding edge between vertex 1 and 2 in graph 0.

VERBOSE ->2017-04-30 18:02:54 :: Found vertex 1 in graph 0.

  ERROR ->2017-04-30 18:02:54 :: Failed to find vertex 3 in graph 0.

VERBOSE ->2017-04-30 18:02:54 :: In graph 0, adding vertex with id 3.

VERBOSE ->2017-04-30 18:02:54 :: Adding edge between vertex 1 and 3 in graph 0.

VERBOSE ->2017-04-30 18:02:57 :: Found vertex 3 in graph 0.

VERBOSE ->2017-04-30 18:02:57 :: Found vertex 2 in graph 0.

VERBOSE ->2017-04-30 18:02:57 :: Adding edge between vertex 3 and 2 in graph 0.

  ERROR ->2017-04-30 18:03:01 :: Failed to find vertex 5 in graph 0.

VERBOSE ->2017-04-30 18:03:01 :: Found vertex 2 in graph 0.

VERBOSE ->2017-04-30 18:03:01 :: In graph 0, adding vertex with id 5.

VERBOSE ->2017-04-30 18:03:01 :: Adding edge between vertex 5 and 2 in graph 0.

  ERROR ->2017-04-30 18:03:08 :: Failed to find vertex 8 in graph 0.

  ERROR ->2017-04-30 18:03:08 :: Failed to find vertex 9 in graph 0.

  ERROR ->2017-04-30 18:03:08 :: Missing vertexId 8 and 9 in graph 0. Edge between the vertices not created.

VERBOSE ->2017-04-30 18:03:09 :: Shutting down database!

3. **Creating schema for an edge or a vertex:**

The logger will keep track of schema creation of edges and vertices. On creation of schema, the logger will update the log showing the previous schema as well as the current one. If there is no previous schema, the logger will log it as ‘null’ in the log. If the schema cannot be added because there is no vertex or edge present, the logger will keep track of that too. Following is example of how the logger keeps track of schema changes in the edge and the vertex.

i. When the edge or vertex is missing,

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g s

grdb> g s v int j

grdb> g s e int k

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_16\_25\_08.txt

VERBOSE ->2017-04-30 16:25:08 :: Database Initialization!

  ERROR ->2017-04-30 16:25:16 :: Error adding schema. Either the vertex or the edge is missing.

  ERROR ->2017-04-30 16:25:24 :: Error adding schema. Either the vertex or the edge is missing.

VERBOSE ->2017-04-30 16:25:30 :: Shutting down database!

DEN-L-TDR53:cli Ranabhat$

ii. Adding schema to the vertex:

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> g s v int i

grdb> g s v int j

grdb> g s v float k

grdb> g s

>graph 0

Sv = [INT:i,INT:j,FLOAT:k]

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_16\_27\_02.txt

VERBOSE ->2017-04-30 16:27:02 :: Database Initialization!

VERBOSE ->2017-04-30 16:27:03 :: Adding graph with id 0.

VERBOSE ->2017-04-30 16:27:03 :: In graph 0, adding vertex with id 1.

VERBOSE ->2017-04-30 16:27:10 :: In graph 0, change vertex schema from null to [INT:i]

VERBOSE ->2017-04-30 16:27:16 :: In graph 0, change vertex schema from [INT:i] to [INT:i,INT:j]

VERBOSE ->2017-04-30 16:27:19 :: In graph 0, change vertex schema from [INT:i,INT:j] to [INT:i,INT:j,FLOAT:k]

VERBOSE ->2017-04-30 16:27:27 :: Shutting down database!

DEN-L-TDR53:cli Ranabhat$

iii. Adding schema to the edge:

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> g e 1 2

grdb> g s e int i

grdb> g s e int k

grdb> g s e double j

grdb> g s

>graph 0

Se = [INT:i,INT:k,DOUBLE:j]

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_16\_29\_56.txt

VERBOSE ->2017-04-30 16:29:56 :: Database Initialization!

VERBOSE ->2017-04-30 16:29:58 :: Adding graph with id 0.

VERBOSE ->2017-04-30 16:29:58 :: In graph 0, adding vertex with id 1.

VERBOSE ->2017-04-30 16:30:01 :: Found vertex 1 in graph 0.

  ERROR ->2017-04-30 16:30:01 :: Failed to find vertex 2 in graph 0.

VERBOSE ->2017-04-30 16:30:01 :: In graph 0, adding vertex with id 2.

VERBOSE ->2017-04-30 16:30:01 :: Adding edge between vertex 1 and 2 in graph 0.

VERBOSE ->2017-04-30 16:30:04 :: In graph 0, change edge schema from null to [INT:i]

VERBOSE ->2017-04-30 16:30:06 :: In graph 0, change edge schema from [INT:i] to [INT:i,INT:k]

VERBOSE ->2017-04-30 16:30:11 :: In graph 0, change edge schema from [INT:i,INT:k] to [INT:i,INT:k,DOUBLE:j]

VERBOSE ->2017-04-30 16:30:14 :: Shutting down database!

DEN-L-TDR53:cli Ranabhat$

4. **Creating enum:**

The log keeps track of the enumerated types that can be used as attributes for vertices and edges. The logger with keep track of attempts to create enums with illegal syntax as well as when duplicate enums are created.

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> enum family \_father mother

Enum element \_father illegal syntax

grdb> enum family father \_mother

Enum element \_mother illegal syntax

grdb> enum family father mother

grdb> enum family father mother

enum family already exists

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_16\_46\_02.txt

VERBOSE ->2017-04-30 16:46:02 :: Database Initialization!

VERBOSE ->2017-04-30 16:46:06 :: Adding graph with id 0.

VERBOSE ->2017-04-30 16:46:06 :: In graph 0, adding vertex with id 1.

  ERROR ->2017-04-30 16:46:14 :: Illegal syntax to create enum: \_father

  ERROR ->2017-04-30 16:46:21 :: Illegal syntax to create enum: \_mother

VERBOSE ->2017-04-30 16:46:25 :: Enum added: family([father mother]).

  ERROR ->2017-04-30 16:46:29 :: Enum [family] already exists.

VERBOSE ->2017-04-30 16:46:31 :: Shutting down database!

DEN-L-TDR53:cli Ranabhat$

5. **Creating tuples for an edge or a vertex:**

The logger keeps track of adding tuples to an edge or a vertex. In absence of a vertex or an edge, it logs the identifying information about the edge or the vertex and logs the inability to update/create the tuple. If unable to update the tuple because of mismatch in schema, the log updates that information. The following example shows the following scenario:

1. In absence of vertex schema, attempt to add a tuple.
2. With correct vertex schema information, add a tuple to absent vertex.
3. With incorrect vertex schema information, add a tuple to present vertex.
4. With correct vertex schema information, add a tuple to present vertex.

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> g t 2 i 10

Missing vertex schema

grdb> g s v int i

grdb> g t 2 i 10

Illegal vertex id

grdb> g t 1 k 10

offset of k not found

Set vertex tuple value failed

grdb> g t 1 i 10

grdb> g t

({1[10]},{})

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_17\_17\_34.txt

VERBOSE ->2017-04-30 17:17:34 :: Database Initialization!

VERBOSE ->2017-04-30 17:17:36 :: Adding graph with id 0.

VERBOSE ->2017-04-30 17:17:36 :: In graph 0, adding vertex with id 1.

  ERROR ->2017-04-30 17:17:39 :: Missing vertex schema. Cannot update the tuple.

VERBOSE ->2017-04-30 17:17:43 :: In graph 0, change vertex schema from null to [INT:i]

  ERROR ->2017-04-30 17:17:47 :: Failed to find vertex 2 in graph 0.

  ERROR ->2017-04-30 17:17:47 :: Missing vertex id 2. Cannot update the tuple.

VERBOSE ->2017-04-30 17:17:52 :: Found vertex 1 in graph 0.

  ERROR ->2017-04-30 17:17:52 :: Error while setting tuple. Tuple entry [10:k] doesn't match vertex schema [[INT:i]].

VERBOSE ->2017-04-30 17:17:57 :: Found vertex 1 in graph 0.

VERBOSE ->2017-04-30 17:17:57 :: In vertex id 1 of graph 0, setting attribute i to 10 in the vertex schema [INT:i].

VERBOSE ->2017-04-30 17:18:01 :: Shutting down database!

DEN-L-TDR53:cli Ranabhat$

The log will also track the following events for updating edge tuples:

1. In absence of edge schema, attempt to add a tuple.
2. With correct edge schema information, add a tuple to present edge.
3. With correct edge schema information, add a tuple to absent edge.
4. With incorrect edge schema information, add a tuple to present edge.

DEN-L-TDR53:cli Ranabhat$ ./db

Graph Database

(C) Frank W. Miller

grdb> g n

grdb> g t

({1},{})

grdb> g t 1 2 i 10

Missing edge schema

grdb> g e 1 2

grdb> g s e float i

grdb> g t 1 2 i 10

grdb> g t 3 4 i 10

Illegal vertex id(s)

grdb> g t 1 3 i 10

Illegal vertex id(s)

grdb> g t 1 2 j 10

offset of j not found

Set edge tuple value failed

grdb> q

DEN-L-TDR53:cli Ranabhat$ cat logfile\_2017\_04\_30\_17\_26\_28.txt

VERBOSE ->2017-04-30 17:26:28 :: Database Initialization!

VERBOSE ->2017-04-30 17:26:31 :: Adding graph with id 0.

VERBOSE ->2017-04-30 17:26:31 :: In graph 0, adding vertex with id 1.

  ERROR ->2017-04-30 17:26:41 :: Missing edge schema. Cannot update the tuple.

VERBOSE ->2017-04-30 17:26:47 :: Found vertex 1 in graph 0.

  ERROR ->2017-04-30 17:26:47 :: Failed to find vertex 2 in graph 0.

VERBOSE ->2017-04-30 17:26:47 :: In graph 0, adding vertex with id 2.

VERBOSE ->2017-04-30 17:26:47 :: Adding edge between vertex 1 and 2 in graph 0.

VERBOSE ->2017-04-30 17:26:54 :: In graph 0, change edge schema from null to [FLOAT:i]

VERBOSE ->2017-04-30 17:27:01 :: Found edge between vertex 1 and 2 in graph 0.

VERBOSE ->2017-04-30 17:27:01 :: Between vertices 1 and 2 of graph 0, setting attribute i to 10 in the edge schema [FLOAT:i].

  ERROR ->2017-04-30 17:27:07 :: Failed to find edge between vertex 3 and 4 in graph 0.

  ERROR ->2017-04-30 17:27:07 :: Cannot update/create edge tuple. Missing vertices 3 and 4 from the graph 0.

  ERROR ->2017-04-30 17:27:11 :: Failed to find edge between vertex 1 and 3 in graph 0.

  ERROR ->2017-04-30 17:27:11 :: Cannot update/create edge tuple. Missing vertices 1 and 3 from the graph 0.

VERBOSE ->2017-04-30 17:27:17 :: Found edge between vertex 1 and 2 in graph 0.

  ERROR ->2017-04-30 17:27:17 :: Error while setting tuple. Tuple entry [10:j] doesn't match vertex schema [[FLOAT:i]].

VERBOSE ->2017-04-30 17:27:20 :: Shutting down database!

DEN-L-TDR53:cli Ranabhat$

NOTE: The code for the logger can be found at the following github link:

<https://github.com/rabin2360/Database_Systems>