**GDBMS**

Project's doxygen: Graph database management system

Main characteristics:

1.

Fuly transactional

2.

Supports formalism called labeled directed attributed hypergraph

3.

Users management, sessions, transactions are separated by the transport system, so that different transport system can be used in future

4.

Its own procedural language for definition of triggers and user scripts

5.

Type definition subsystem, which supports class object-like hierarchy structures, arrays definitions, simple types aliases definitions

GDBMS is designed and implemented to support Graph Database formalism called Directed, labeled hyper-graph, which increases the power of the set of applicable tasks which can be solved by such a system. More efficient disk usage is also achieved by using the formalism: 2 or more edges can be represented as 1 hyper-edge. Another important feature of the system is that each of the nodes and edges can be of different user types. User types can be user defined structures, arrays or simple type aliases for integers or strings. The system is transactional, user authentication and authorization, sessions and transactions management. The transactional subsystem is quite sophisticated unlike the transactional subsystems of other lookalike graph database systems. It insures that only those edges or nodes, which have been added or altered by the transaction are in the transactional folder and/or locked by the transaction. Information about the transaction history is maintained, and when transaction close command is received, some edge/node operations dependencies are resolved and handled: Insert after update, update after insert, update after update, for instance. Data is stored in xml based hash table, which balances the relative complexity of search operations and insert and alter operations and achieves easier transactional lock of files when needed. If binary trees were used, much effort on balancing the trees would be required, when insert and update operations occur.The hybrid data model which has relational and hierarchical characteristics is also easily achieved by using multiple hash tables.

Search and iteration operations support limiting the number of results when too big graphs are iterated.

Extreme path queries support relative path user specification to a valid value in a valid type path in the type hierarchy for a node.

The metric in terms to the extreme path queries is a valid type path to a numeric value in each edge, which is used by Dijkstra algorithms to determine which of the paths is longest or shortest. The metric can be determined in a user trigger, specified in a user defined script by a formula or directly, depending on the user application.

The web service interface remote methods, which directly support call of each operations is obsolete and the user is encouraged to use the methods, which allow script interpretation and user script triggers registration and interpretation.

As a conclusion we can claim that the system has everything needed to serve as a basis of a numerous set of applications, which are graph based or hypergraph oriented: routing protocols simulations; map tasks; social networks graphs; Set oriented tasks (Hyperedges can be viewed as a sets of nodes), etc.

The user is able to define its own event handlers. The language is procedural. It supports its own flow identifier operator for graphflow queries for nodes and edges, which is used in combination with foreach operator.