Claims:

1. A verifiable ledger database configured to manage transactions of a ledger, comprising a plurality of shards, each shard being formed by partitioning transaction data and comprising:

a ledger storage configured to provide access to the transaction data and support a plurality of proofs;

a transaction manager configured to execute each transaction according to a respective transaction request; and

a verifier configured to verify the transactions by returning the proofs according to verification requests, and

one or more auditors for ensuring correct execution of the verifiable ledger database server, wherein each auditor is configured to:

check whether the verifiable ledger database server forks the history log by checking whether users of the verifiable ledger database receive digests that correspond to a linear history; and

re-execute the transactions to ensure that current states of the verifiable ledger database are correct.

2. The verifiable ledger database of claim 1, wherein the ledger storage comprises an upper level pattern-oriented split tree (POS-tree) and a lower level POS-tree, the upper level POS-tree and the lower level POS-tree each comprises a plurality of nodes including at least one root node and at least one leaf node, and wherein the ledger storage comprises a hash chained sequence of blocks and each leaf node of the upper level POS-tree comprises a block number for a respective block.

3. The verifiable ledger database of claim 2, wherein the lower level POS-tree is built on states of the verifiable ledger database, and the lower level POS-tree comprises one or more root nodes each containing a respective root hash and a respective said block number.

4. The verifiable ledger database of claim 3, wherein the root nodes of the lower level POS-tree are stored as leaf nodes of the upper level POS-tree.

5. The verifiable ledger database of claim 4 being configured to retrieve the transaction data from a given block number by locating a corresponding leaf node of the upper level POS-tree with the given block number and then traversing the lower level POS-tree to locate the transaction data.

6. The verifiable ledger database of claim 4, wherein the verifiable ledger database is configured to update the transaction data by creating new nodes at the lower level POS-tree and the upper level POS-tree using copy-on-write.

7. The verifiable ledger database of any one of claims 1 to 6, wherein the transaction data is modelled as a plurality of keys, the verifiable ledger database is configured to partition the keys into the shards based on the hash of the keys.

8. The verifiable ledger database of claim 7 being configured to use a two-phase commit (2PC) protocol to ensure atomicity of the transactions.

9. The verifiable ledger database of any one of claims 1 to 8, wherein the transaction manager is configured to log each transaction and respond with a commit or abort based on a concurrency control algorithm.

10. The verifiable ledger database of any one of claims 1 to 9, wherein the transaction manager comprises:

    a transaction queue for buffering the transaction requests;

    a plurality of transaction threads for receiving the transaction requests buffered in the transaction queue;

    a shared memory for storing prepared transactions and committed transaction data; and

    a persisting thread for persisting the committed transaction data asynchronously to the ledger storage.

11. The verifiable ledger database of claim 10, wherein the transaction manager is configured to execute the transactions by:

    assigning the transaction requests buffered in the transaction queue sequentially to the transaction threads if the transaction queue is not full; and

aborting the transactions if the transaction queue is full.

12. The verifiable ledger database of claim 10 or 11, wherein the transaction manager is configured to update ledger stored in the verifiable ledger database asynchronously by:

allowing the shared memory to store the transaction data in a committed data map when the transaction manager receives a commit message;

writing to a write-ahead-log (WAL);

allowing the persisting thread to persist the transaction data in the committed data map to the ledger storage to generate persisted data; and

removing the persisted data from the committed data map.

13. The verifiable ledger database of claim 12, wherein the transaction manager is configured to batch the transactions to be committed before updating the ledger by:

collecting respective data from each transaction to be committed into a data block; and

appending the data block that are created within a time window to the ledger storage.

14. The verifiable ledger database of claim 13, wherein collecting the respective data from each transaction to be committed comprises selecting the respective data from the committed data map version by version.

15. The verifiable ledger database of any one of claims 1 to 14, wherein the proofs comprise one or more of an inclusion proof, a current-value proof, and an append-only proof.

16. The verifiable ledger database of any one of claims 1 to 15, wherein the verifier is configured to batch the proofs for the transaction data.

17. The verifiable ledger database of any one of claims 12 to 16, wherein the verifier is configured to return the proofs of the persisted data immediately during transaction processing.

18. The verifiable ledger database of any one of claims 1 to 17, wherein the verifier is configured to verify the transactions within a time window after the transaction is processed.

19. The verifiable ledger database of any one of claims 1 to 17, wherein the verifier is configured to verify read set and write set of each transaction.

20. The verifiable ledger database of any one of claims 2 to 19, after one or more nodes crash and reboot, being configured to recover said one or more nodes.