What is a database ledger

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:35 AM PDT

Contents

What is a database ledger

The database ledger logically leverages a blockchain and merkle tree data structures. The database ledger incrementally captures the state of the database as it evolves over time while updates occur on Ledger tables. To achieve that, the Database Ledger stores an entry for every transaction capturing metadata about the transaction, such as its commit timestamp and the identity of the user that executed it, but also the Merkle Tree root of the rows updated in each Ledger table. These entries are then appended to a tamper-evident data structure to allow verifying their integrity in the future.

To get a better understanding on merkle tree data structure read the references in theWhitepaper

The data regarding transactions and blocks is physically stored as rows in two new system catalog views:

1. **sys.database_ledger_transactions** - maintains a row with the information of each transaction in the ledger, including the ID of the block where this transaction belongs and the ordinal of the transaction within the block.
2. **sys.database_ledger_blocks** - maintains a row for every block in the ledger, including the root of the Merkle tree over the transactions within the block, as well as the hash of the previous block to form a blockchain.

sys.database_ledger_transactions schema

Column name	Data type	Description
transaction_id	bigint	A transaction id that is unique for the database (it corresponds to a transaction id in the database transaction log).
block_id	bigint	A sequence number identifying a row.
transactional_ordinal	int	Offset of the transaction in the block.
user_name()	sysname)	The name of the user who initiated the transaction. Captured by calling ORIGINAL_LOGIN().
commit_time	datetime2(7)	The time of the committing transaction.
table_hashes	varbinary(max)	This is a set of key-values pairs, stored in a binary format. The keys are object ids (from sys.objects) of ledger database tables, modified by the transaction. Each value is a SHA-256 hash of all row versions a transaction created or invalidated. <version> - indicates the encoding version. Length: 1 byte. <length> - the number of entries in the key-value pair list. Length: 1 byte.<key> - an object id. Length: 4 bytes. <value> - the hash of rows the transaction tached in the table with the object id stored as the key. Length: 32 bytes. |

sys.database_ledger_blocks schema

Column name	Data type	Description
block_id	bigint	A sequence number identifying the row in this view.
transaction_root_hash	binary(32)	The hash of the root of the Merkle tree, formed by transactions stored in the block.
block_size	bigint	The number of transactions in the block.
previous_block_hash	binary(32)	A SHA-256 hash of the previous row in the view.

To view the database ledger, execute the following T-SQL statements in SQL Server Management Studio or Azure Data Studio.

```
SELECT * FROM sys.database_ledger_transactions
GO

SELECT * FROM sys.database_ledger_blocks
GO
```

A block is closed every 30 seconds, or when the user manually generates a database digest through executing the **sys.sp_generate_database_ledger_digest** stored procedure. When a block is closed, new transactions will be inserted in a new block. The block generation process then retrieves all transactions that belong to the “closed” block from both the in-memory queue and the sys.database_ledger_transactions system table, computes the Merkle tree root over these transactions and the hash of the previous block and persists the closed block in the sys.database_ledger_blocks system table. Since this is a regular table update, its durability is automatically guaranteed by the system. To maintain the single chain of blocks, this operation is single-threaded, but it is also very efficient, as it only computes the hashes over the transaction information, and happens asynchronously, thus, not impacting the transaction performance.

How good have you found this content?