Hekaton introduces a  new database engine introduced for OLTP workloads. Since, memory has become cheaper in recent times, it has become convenient to store whole data in memory. Thus, this has eased the OLTP (Online Transaction Processing). Different. Now, these OLTP databases which can be as huge as 1 TB (Terabyte) can be loaded into the memory. It is an integrated engine in the SQL Server,  rather than a standalone database system. Traditionally different database systems have been loaded separately for solving OLTP needs, which is an issue in handling different database systems, for the developer or user.

Hekaton tables are stored in the memory, and are accessed using the T-SQL. Also, transactions can update both the T-SQL and the normal SQL tables Heakton also has T-SQL stored procedures. Users declare a table as memory optimized, following which the Hekaton stores the whole table in the memory. These tables are stored as a snapshot in the memory.

Different components in the Hekaton architecture are: Storage engine, compiler, query processing, runtime system, transactions, storage and log.
Hekaton tables have different indexes: hashes and range indexes, implemented by lock-free hash tables and B-trees. Having multiple indexes on a single table, each of the read operations or lookup is done using the indexes on the table. Also, having the lock free hashes allows access to the data concurrently.  Checkpoints are created in the logs for transaction, to check its commit time, and also in case of recovery.

Strong  Points:

- Since, Hekaton is integrated within the MS SQL Server, there is no need to manage different databases for the OLTP workload.
- Having lockfree hashes, results in fast and concurrent access to the data.
- The threads also check the commit time for any transaction, while accessing it, to check if the transaction is unmodified while it was being accessed.

Weak Points:

- Since Heakton has multiple versions of the snapshot in the memory for the database, there is a need to garbage collect the different snapshots lying in the memory. These snapshots include older versions of the database.
- Garbage collection is done in the background or by the query scan processes, thus, in the data tables where the query scan is not running, might not be garbage collected, or at times, different versions can be cleared which are in need for the query.

Question:

- Hekaton uses group committing for transactions, so what happens if one set of transactions are not logged, as these transactions are not yet committed due to the group commit feature, and the database fails? From what point does the recovery happen?