Performance

HyperSQL has multiple deployment and persistence options which influence its performance.

1. The table type, MEMORY, CACHED or TEXT, indicates how the table row data is stored and accessed by the database engine.
2. In-process or server mode indicates how all the database engine data is accessed by the user's application.
3. The transaction model indicates how and when different sessions (connections) wait for one another.
4. Use of BLOBS and CLOBS

Table Type

The performance characteristics of different table types are:

1. MEMORY tables provide the highest performance. All data is in memory and each field of each row is a memory object that can be read by the database engine without any conversion. When data is updated, only a log record is written to disk, usually with a fixed overhead per row.
2. CACHED tables have a lower performance compared to MEMORY tables. Data for this type of table comes from a row cache (hence the name) which at any time holds a subset of all the rows in all CACHED tables. Reduction in performance is due to three reasons.
    - First, because the size of the row cache is usually smaller than the total row count of all the CACHED tables, rows are frequently purged from the cache and other rows are read from the disk and converted into memory objects.
    - Second, even if the row cache is large enough to hold all the rows that are accessed by the user's application in a period of time, the extra layer of access adds a small overhead.
    - Third, because data updates are both logged and written to the data file, there is a larger overhead for data updates.
3. TEXT tables have similarities to both CACHED tables and MEMORY tables. The indexes are held in memory, while the data is held on disk and cached like CACHED tables. Because the data is stored in text form as comma separated values (CSV) or similar formats, reading and writing the data takes longer than the same operation in binary format. On the other hand, because the indexes are in memory and no separate logging is performed, write operations may be faster than CACHED tables.

In-process and Server

The performance characteristics of in-process versus server mode are:

1. In-process access takes place in the same memory space as the user's application. There is no data conversion or network transfer overhead involved.
2. Server mode usually has a different memory space than the user's application. The data is converted into a byte stream, transferred over the network, and then converted back into objects. This introduces latency plus the extra processing needed for conversion.
3. HyperSQL supports SQL and Java stored procedures, which allow a whole transaction to be encapsulated in a single SQL statement. This speeds up access in the server mode as a transaction can be completed in a single network round trip, instead of execution of several statements. A single call to a stored procedure can even return multiple result sets and return values to the user's application.

Transaction Model

HyperSQL supports MVCC and two phased locking transaction models. The performance characteristics of the transaction model are:

1. HyperSQL is fully multi threaded. If the vast majority of operations are read operations, then performance is very high in all transaction models. Multiple processes, each running in a different thread, can access the same tables or rows at the same time and return the results independently to the user's application(s).
2. If there is a significant amount of update operations, the 2PL lock model performance can be reduced to a single thread. Because locks are kept both for reads and writes, read operations on an updated table are delayed until the writer commits, and similarly write operations are delayed until the reader commits.
3. The multiversion concurrency control (MVCC) model provides vastly greater performance than the lock based model, as no read locks are used, while write locks are kept only on the individual updated rows. Multiple threads can read and update the database using multiple processor cores. This mode is more performant than row level locking modes supported by some other database engine.
4. With MVCC and multi processor cores, if there is spare processing power and many concurrent sessions, the time overhead of network communications will not affect the overall performance of the server mode deployments. Each connection runs in a separate thread and uses the available processing power for data conversion.

In summary, the fastest performance is typically achieved with the combination of **MEMORY tables**, **in process access** and **MVCC transaction model**. If reduced memory use is required, some tables can be defined as CACHED tables, while keeping the most frequently accessed tables as MEMORY tables. If **server access** is required, then **stored procedures** can be used to reduce the network round trips.

Blobs and Clobs

HSQLDB is the only SQL open source database that supports a dedicated LOB store. Blobs and clobs can be very large and benefit from a separate store that avoids mixing their data with row data which is not too large. Internal database tables are used for the LOB catalog. Therefore each access to a LOB has the overhead of catalog lookup. This overhead is justified when the stored LOBs are large. HSQLDB supports long VARCHAR and VARBINARY columns that can be used instead of CLOB and BLOB especially when the average lob size is below 32 KB. These types do not have the LOB catalog overhead.

See the Performance Tests page for some benchmark test results.