



WHITE PAPER

Performance & Tuning for In-Memory Databases in Adaptive Server Enterprise 15.5

Contents

1. Introduction	2
IMDB & ACID.....	3
Benefits of ASE In-Memory Databases	3
Use Cases for ASE In-Memory Databases.....	4
ASE 15.5 Syntax Details.....	4
2. Internal ASE Optimizations for IMDBs and RDDBs.....	5
3. How to Utilize IMDBs and RDDBs in Applications.....	7
4. IMDBs: performance test results	8
Beta customer test results.....	8
Sybase Internal Test Results	8
Scenario: Transaction Processing	9
Scenario: Processing Streaming Market Data.....	10
Scenario: Telecom Call Processing.....	11
Scenario: Various Types of Batch DML	12
5. Using Minimally Logged DML (ML-DML).....	13
ML-DML Internals	13
Enabling ML -DML.....	14
Restrictions for using ML-DML.....	14
Determining whether ML-DML is used.....	15
ML-DML: Impact on transactional semantics	16
Tuning ML-DML	16
ML-DML : performance test results.....	17
6. The ASE Query Optimizer	20
Showplan Changes.....	20
Compatibility Mode and IMDBs/RDDBs	21
ML-DML and the Query Optimizer	21
7. Performance Tuning: Moving from Bottleneck to Bottleneck.....	22
Tuning for IMDBs and RDDBs	22
Tuning for RDDBs.....	23
Tuning for Minimally Logged DML.....	23
8. Using IMDBs as Temporary Databases	24
Using RDDBs as temporary databases.....	24
Using tempdb groups.....	25
9. Further Reading.....	27
10. Summary	28

Revision History

Version 1.0 - January 2010

Version 1.1 - February 2010

1. Introduction

Version 15.5 of Sybase Adaptive Server Enterprise (ASE) provides In-Memory Database (IMDB) capabilities designed to deliver low response times and high throughput for mission-critical systems. More specifically, IMDBs tend to deliver better performance for write-intensive workloads. This white paper looks at performance aspects of this ASE option.

IMDB & ACID

Traditionally, ASE databases are designed for applications that need to strictly adhere to ACID transaction semantics (ACID is short for: Atomic, Consistent, Isolated, Durable). These ACID properties are implemented by means of a write-ahead transaction log, located on persistent storage (i.e. disk). However, many business applications contain elements that may not require full ACID semantics and could accept a compromise on some of these properties in order to achieve performance gains.

Against this background, the IMDB feature in ASE 15.5 allows the 'Durable' and 'Atomic' aspects of transactional behavior to be relaxed in exchange for lower response times and higher throughput. This is achieved by partly or completely omitting transaction logging, and by avoiding persistent disk storage.

The ASE 15.5 IMDB feature introduces two new types of ASE databases:

- The ASE In-Memory Database (IMDB) is a zero-disk footprint database optimized to run completely in memory, and has no associated disk storage, and consequently, no overheads due to disk I/O. Transaction logging is still performed to support run-time rollbacks, and other operations like triggers and replication, but is done entirely in-memory.
- The ASE 'Relaxed-Durability Database' (RDDB) is stored on disk as regular ASE databases, but has many of the same performance optimizations as IMDB databases. RDDB databases can be used in situations where the database size is too large to fit entirely into the available memory and the applications require higher performance in exchange for relaxed durability.

ASE 15.5 introduces a new database property named 'durability', referring to the 'D' in ACID. For an IMDB database, durability will always be set to `no_recovery`, indicating that committed transactions will be lost after a restart of ASE. For an RDDB database, the durability can be one of `no_recovery` or `at_shutdown`, the latter indicating that transaction durability is ensured only after a normal (polite) ASE shutdown. Regular user databases with full ACID support and system databases always have a durability setting of 'full'.

Databases with a durability setting of `no_recovery` will be recreated from scratch, typically the `model` database, when ASE is restarted. Optionally, a regular user database can be configured as a template for such databases, instead of using the default template of the `model` database.

To relax the Atomic aspect of transactions (the 'A' in ACID), ASE 15.5 allows minimally logged DML to be performed in IMDB and RDDB databases.

Benefits of ASE In-Memory Databases

It is beyond discussion that certain applications will always require full ACID transactions; for example a system handling bank transfers should be able to rely on transactions that are both Atomic and Durable.

In addition to supporting full ACID properties, ASE 15.5 will offer customers more choices to relax ACID properties in exchange for better performance.

An ASE IMDB or RDDB adds no complexity to the overall system architecture and operational environment. Unlike products from other vendors, ASE's in-memory database is not an additional software component that must be separately installed, configured, started and monitored. Instead, an ASE IMDB or RDDB is managed by ASE and can be used by applications in much the same way as any other ASE database.

Use Cases for ASE In-Memory Databases

ASE In-Memory databases with relaxed ACID properties can be useful in many situations, such as the following:

- Data-intensive business applications can benefit from the overall performance advantages of ASE IMDB technology. For example, trading systems could allow reference and compliance data to be cached in an IMDB to provide low-latency and high concurrency access to read-mostly data. Also, data from multiple sources can be cached in an IMDB for locality of reference and quick look-ups, and in-memory data can be kept constantly synchronized as source data changes.
- ASE IMDB capabilities can be used to process massive streams of incoming data such as market data feeds, news feeds and sensor data feeds by supporting high-speed inserts and highly concurrent delete/insert/update activity with minimal overhead.
- Monitoring systems with real-time dashboards benefit from real-time views of business critical systems and the removal of highly repetitive read activity from core business systems.
- For large batch processes that take derived data as input, ASE IMDB offers improved performance. Since the original data sources are not being modified, ACID properties can be relaxed without risk of losing source data in case of a failure.
- ASE IMDB is ideally suited for applications where data needs to be stored only for a short period of time. For example, in an E-commerce application, a customer's shopping cart does typically not need to be kept indefinitely, and could well be stored in an ASE IMDB (in contrast, once the shopper completes and pays his order, the resulting transactions must be handled with full ACID properties).

ASE IMDB provides a diskless alternative for applications performing queries that make intensive use of the ASE temporary database. By creating an in-memory temporary database, overall performance improvements can be realized.

ASE 15.5 Syntax Details

This whitepaper covers only performance aspects of IMDBs and RDDBs. For full syntax details on the new ASE 15.5 feature, please refer to the ASE 15.5 documentation set (also see page 27).

2. Internal ASE Optimizations for IMDBs and RDBBs

In-memory databases (IMDBs) and Relaxed-durability databases (RDBBs) are normal databases from the perspective of SQL functionality and client connectivity. However, they differ from regular user databases in a number of ways, mostly related to aspects of transaction logging and disk I/O. IMDBs and RDBBs have been optimized in these areas to deliver better performance, in exchange for relaxing certain aspects of transactional semantics.

These optimizations are internal to ASE and cannot be individually disabled (they are enabled by default). As background information, a short overview of the various optimization aspects is presented below. Note that none of these optimizations can be adjusted, tuned or disabled: they are always active for IMDBs and RDBBs.

- **IMDBs do not perform disk I/O and therefore avoid certain aspects of buffer management.**
Since an IMDB runs entirely in memory, no disk I/O needs to be performed (an exception is the disk I/O to read the contents of the template database or 'model' when the IMDB is created. But since this is a one-time operation only, it doesn't count for the purpose of this discussion). The absence of disk I/O makes some aspects of buffer management in ASE caches redundant. For example, the `inmemory_storage` caches that are required for an IMDB do not maintain an LRU/MRU chain of pages, and the concepts of large I/O buffer pools, 'wash marker', asynchronous prefetch, and buffer replacement policy do not apply. Consequently, the memory structures for such a cache need to be modified much less than for classic named caches, allowing better performance to be delivered.
For RDBBs, which still perform disk I/O, the above optimizations do not apply and these concepts are still valid.
- **Optimizations around write-ahead transaction logging.**
Because IMDBs and RDBBs do not need to guarantee recoverability of the database under all circumstances, some aspects of ASE's write-ahead transaction logging mechanism have been optimized. Among these are internals known as 'log pinning' and 'ULC pinning'. For both, transaction log records must be written to the transaction log immediately when a certain order of page modifications occurs, in order to satisfy the rules of ASE's write-ahead transaction logging. This write-ahead logging mechanism guarantees that if ASE were to suddenly shut down in the middle of operations (for example, due to a sudden power failure), all completed -that is, committed and rolled back- transactions will be correctly recovered when ASE restarts.
Another aspect of ASE's transaction logging in regular databases is that committing a transaction guarantees that the corresponding log records will have been written to disk. For IMDBs and RDBBs, this guarantee has been removed. For RDBBs, transaction log records may be written to disk at some point in time after the commit, or eventually, for RDBBs with `durability=at_shutdown`, during the normal (polite) ASE shutdown.

Since IMDBs and RDBBs do not require such strict under-all-circumstances recoverability guarantees, these aspects of transaction logging have been done away with for those database types. These optimizations result in less contention around the transaction log, and specifically, the transaction log semaphore. This is beneficial for performance in systems with high transaction rate, since these mechanism translate into additional transaction log access and contribute to performance bottlenecks around the transaction log.

- **Discard log records for completed transactions fully fitting within ULC.**
For IMDBs and RDDBs, a further transaction logging optimization is applied. Whenever a completed transaction fully fits within the session's ULC (User Log Cache), in principle the transaction is discarded completely, and no attempt is even made to flush the log records to the transaction log. This optimization also relieves contention on the transaction log semaphore.
- **Various other internal optimizations.**
Apart from the main optimizations described above, IMDBs and RDDBs come with various other internal optimizations. For details, refer to the chapter on performance and tuning in the In-Memory Database User's Guide which is part of the ASE 15.5. documentation set.

3. How to Utilize IMDBs and RDDBs in Applications

To take full advantage of the capabilities of IMDBs and RDDBs in ASE 15.5, applications should be analyzed to identify queries or tables that can be run or stored in an IMDB or RDDB. This requires that such queries can tolerate the relaxed ACID properties implied by IMDBs or RDDBs.

This analysis may or may not be simple, and will likely be different for each customer and each application. Also, changes to the application may be required to let some queries run in an IMDB. For example, it may be possible for some tables that are currently located in a regular user database to be relocated to an IMDB.

The performance test results described in this document may provide some indications about the types of queries that may benefit from using IMDBs or RDDBs.

One possibility that can always be considered is to create temporary IMDBs, as described on page 24, as this is fully transparent to applications and queries.

4. IMDBs: performance test results

Internal performance tests by Sybase, as well as performance test results by customers participating in the ASE 15.5 beta program, indicate that ASE 15.5 IMDBs may deliver significant performance benefits. A number of different scenarios are described below.

Please keep in mind that, as always, the actual performance results may vary and depend on the specifics of every customer's system and workload.

Beta customer test results

During the beta testing of ASE 15.5, a customer reported overall performance gains of up to a factor 10. This result was achieved when running part of an existing securities trading application against an IMDB in ASE 15.5.

This particular part of the application consisted of stored procedures performing pre-trading compliance checks. The average performance improvement of factor 9-10 was achieved by deploying the ASE 15.5 and relocating data into the IMDB; no other changes to schema or SQL code were made.

For transaction processing workload, beta customers reported improvements up to a factor 6; for batch workload (such as data cleansing, summarization), performance gains of factor 2-3 were reported.

Sybase Internal Test Results

Results from internal ASE 15.5 performance tests at Sybase are described below. The results are compared with fully cached, regular (i.e. fully ACID-guaranteed) ASE 15.0.3 disk-resident database. In general, the potential for performance improvement may depend on one or more of the following:

- the type of query or application
- the degree of application concurrency
- the number of ASE engines
- the application's data access patterns

In general, IMDBs tend to deliver better performance for write-intensive workloads.

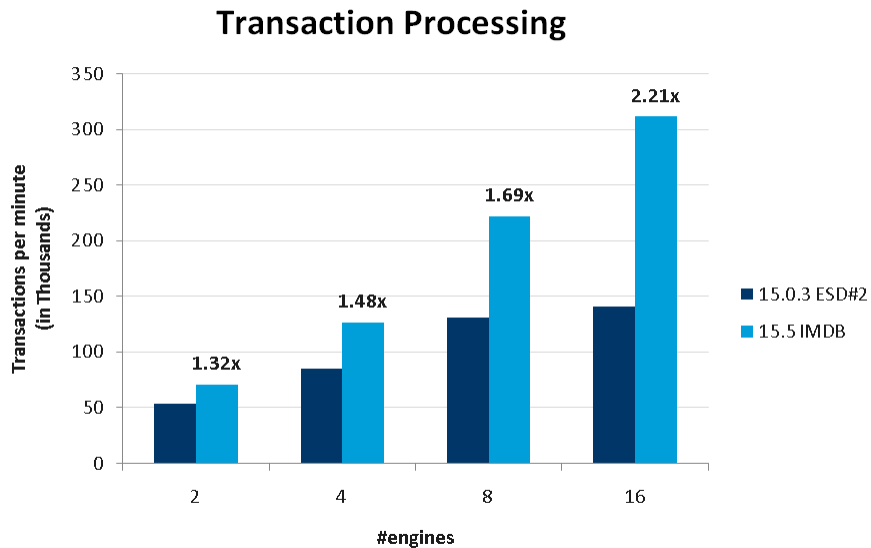
Also, a pattern emerging from testing is that applications using IMDBs scale better than when using classic, full-durability ASE databases. This can be explained by the absence of disk writes in IMDBs, and especially by the reduced contention around the transaction log in an IMDB.

Scenario: Transaction Processing

This test runs a transaction processing workload in an IMDB. The results below showed an ASE 15.5 IMDB to perform 1.3 - 2.2 times better than a fully cached regular ASE database. The IMDB gains improved with increasing concurrency and a higher number of ASE engines.

For this test, a transaction processing workload was used with a workload mix of 33% inserts, 64% updates, and 3% deletes. The tests were conducted with 35 clients for every 4 ASE engines.

Minimally Logged DML was not used in these tests.



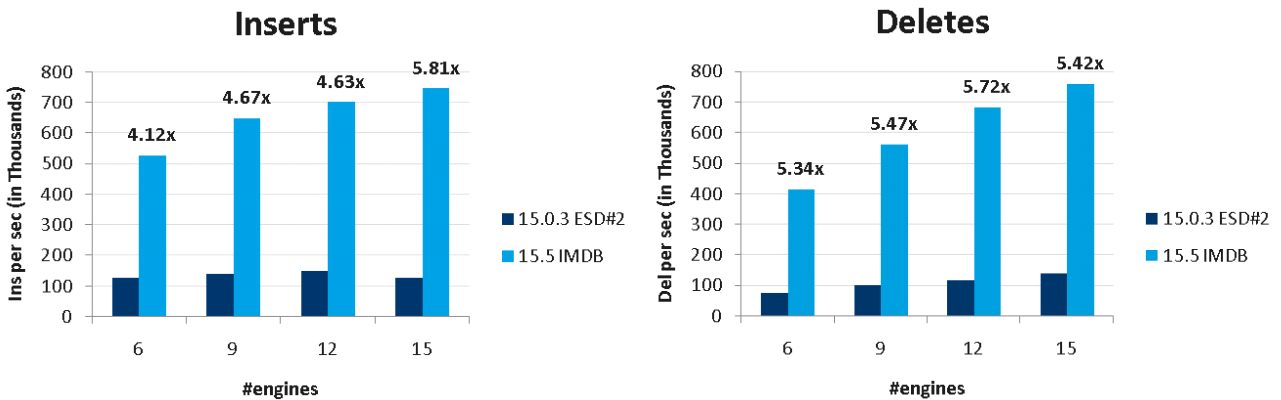
Scenario: Processing Streaming Market Data

The test below involves an application processing a tick stream of incoming financial market data. Three types of client applications, one of each per three ASE engines, concurrently perform the following operations on the same set of IMDB-based database tables:

- bulk-inserting data into round-robin partitioned tables (i.e. inserting 'new' data)
- deleting 'old' data using range deletes
- selecting data

The test results showed an IMDB performing 4 - 6 times better than a fully cached regular ASE database for both inserts and deletes.

Minimally Logged DML was not used in these tests.



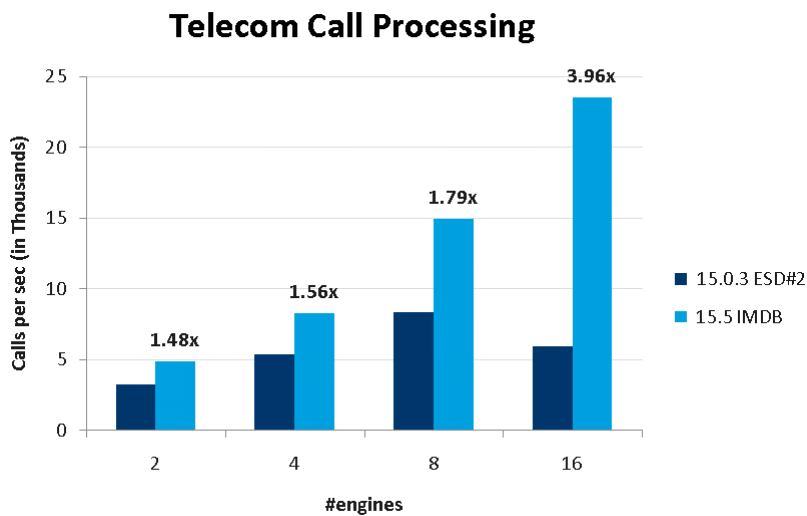
Scenario: Telecom Call Processing

The test below models a write-intensive application like those found in telecom systems, which insert a continuous stream of call details coming in from the telco network, into a database table.

For this test, a workload mix was used of 95% insert+update queries and 5% select queries, with 25 clients per ASE engine.

The test results showed an IMDB performing 1.5 - 4 times better than a fully cached regular ASE database.

Minimally Logged DML was not used in these tests.



Scenario: Various Types of Batch DML

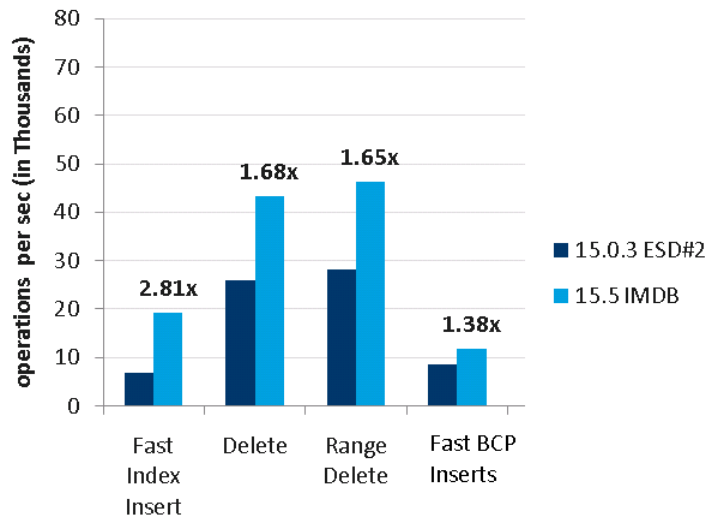
The test below covers different types of DML operations which may be found in applications doing batch processing. One client application is used without any concurrent workload on a 2-engine ASE server.

The table being operated on has a clustered index plus 2 non-clustered indexes.

The test results showed an IMDB performing 1.4 - 2.8 times better than a fully cached regular ASE database.

Minimally Logged DML was not used in these tests.

Selected Types Of Batch DML



5. Using Minimally Logged DML (ML-DML)

ASE 15.5 allows applications to achieve gains by using Minimally Logged DML (ML-DML) in IMDBs and RDDBs. When using ML-DML, ASE attempts to avoid putting log records into the transaction log for `insert`, `update` and `delete` statements, as well as for BCP-in into tables with indexes.

Using ML-DML can bring the following benefits:

- For concurrent DML workload, avoiding transaction logging relieves contention on log semaphore; this applies to both IMDBs and RDDBs.
- In RDDBs, avoiding generation of log records can deliver some performance benefits for an individual session since less log records need to be written to disk. For individual sessions in an IMDB, ML-DML will likely not result in meaningful performance gains since disk writes have been eliminated already in IMDBs.
- In IMDBs and RDDBs, less database space needs to be reserved for the transaction log. This can especially be useful when performing large batch updates which are known to be unlikely to ever be rolled back.

ML-DML is supported on IMDBs and RDDBs, including temporary IMDBs. If the user tries to configure ML-DML in a user database with full durability or in the classic tempdb, it will be silently ignored and full logging will be used. Users can configure ML-DML per database, per table and on session level, as discussed below.

ML-DML Internals

When Minimally Logged DML (ML-DML) is used in IMDBs or RDDBs, ASE will still generate all log records internally, but tries to avoid flushing these log records to the transaction log.

When ML-DML is enabled, ASE does not guarantee the 'Atomic' property of transactions (see the section on transactional semantics on page 16). However, ASE will still guarantee the integrity and consistency of a data row and its corresponding index rows.

For this, when ML-DML is enabled, ASE 15.5 internally uses the notion of a 'subcommand' which conceptually can be thought of as a 'mini-transaction' encompassing all modifications to an individual data row and its corresponding index rows.

When ML-DML is indeed used for a DML statement, and all log records for a subcommand fit in the session's ULC (User Log Cache), they will be discarded when the subcommand has completed successfully. In this case, no log records will be seen in the transaction log. However, when a subcommand does not fully fit in the ULC, it is flushed to the transaction log in its entirety.

When ML-DML is active, it is therefore possible that log records still occur in the transaction log, for example when the ULC size is too small to contain all log records for the average subcommand, or when occasional additional log records increase the size of a subcommand beyond the ULC size (this could happen due to additional log records for index maintenance and space allocation for example). Determining the optimal size for the ULC is therefore an important tuning point when aiming to fully utilize the benefits of ML-DML.

Although subcommands play an important role in the context of ML-DML, note that users cannot usually see evidence of subcommands being used. Subcommands are visible for users only when their log

records occur in the transaction log as a result of exceeding the ULC size (see page 17). Also, users cannot control or influence whether a particular subcommand is 'committed' or 'rolled back' (though these terms are used here only conceptually): this is controlled by ASE only.

Enabling ML -DML

To use ML-DML, this must first be enabled as it is disabled by default. ML-DML can be enabled either on the level of a database, for a individual table, or for a session.

Some syntax examples are shown below (please refer to the IMDB User's Guide for full syntax details):

```
create database my_db set dml_logging = minimal
```

```
alter table my_table set dml_logging = minimal
```

```
set dml_logging minimal
```

As usual in ASE, the session-level setting overrides table-level settings, which in turn override database-level settings.

Apart from enabling the `dml_logging` option as above, the database option `'select into /bulkcopy/pllsort'` must be enabled for ML-DML to be possible.

Note that `set dml_logging` can be used in a login trigger. However, unlike most `set` options, `set dml_logging` cannot be exported from stored procedures or from `execute immediate` to their callers, even when `set export_options` has been enabled.

Restrictions for using ML-DML

A number of additional restrictions apply for actually using ML-DML, as listed below. There will be no error message when fully logged DML is used due to any of these restrictions. The command `set show_exec_info on` can be used to display the actual logging mode.

The restrictions are the following:

- ML-DML can only be used in IMDBs or RDBs. Any attempt to use ML-DML in any other database will be silently ignored.
- ML-DML applies only to user tables and temporary tables; DML on system tables will always be fully logged.
- ML-DML will not be used when a trigger exists for the type of DML being executed; in this case, the DML will always be fully logged so as not to violate any business logic in the trigger. If ML-DML is desired anyway, the DBA must explicitly disable the applicable trigger.
- ML -DML will not be used for tables that participate in Primary Key - Foreign Key constraints (also known as 'references' constraints). For such tables, DML will always be fully logged.
- For tables that are marked for replication, DML will always be fully logged.

- Within one transaction, it is not possible to first perform fully logged DML and then switch to minimally logged DML for the same table in that transaction: in this case, fully logged DML will be used for this table for the rest of the transaction (the reverse order is allowed: first minimally logged DML, then fully logged).
- For query plans that perform a deferred insert, update, or delete, the DML will always be fully logged since deferred access methods must write to the transaction log anyway.
- When a savepoint is created, all DML after the savepoint will be fully logged so as to guarantee the semantics of savepoints.

Occasionally, log records may still be generated even when ML-DML is active; this can happen when the ULC is too small. See 'ML-DML Internals' on page 13 and 'Tuning ML-DML' on page 16.

Determining whether ML-DML is used

Whether ML-DML is actually used is a run-time decision depending on various criteria described above. To determine whether ML-DML is actually used, the command `set show_exec_info on` has been added to ASE 15.5. This command will display the logging mode for every DML command as it is executed:

```
1> set dml_logging minimal
2> go
1> set show_exec_info on
2> go

1> delete imdb1..my_tab
2> go
Operating on the table 'my_tab', database 'imdb1' (owner ID 1) in
'minimal' logging mode by user ID 1.

1> delete regular_db..your_tab
2> go
Operating on the table 'your_tab', database 'regular_db' (owner ID 1)
in 'full' logging mode by user ID 1.
```

Note that `set show_exec_info on` will not display these messages for system tables and for non-IMDB/RDDB databases where full logging will always be used irrespective of any ML-DML setting.

When a transaction is rolled back with the `rollback` command, it will print a message when the transaction contains minimally logged DML operations. This message is informational only:

```
1> rollback
2> go
```

Warning: Only fully logged operations of this transaction will be rolled back. Minimally logged operations will not be rolled back.

ML-DML: Impact on transactional semantics

As DML transactions with minimal logging do not write all log records, it is not possible to fully roll back the changes done in the transaction. A transaction can include changes to different tables, some of which were modified with minimal logging and others with full logging. Upon rollback, only the changes done to tables with full logging will be rolled back, and the changes done to tables with minimal logging will remain committed. When a table is modified in a transaction with minimally logged DML mode, then the atomicity of the transaction is reduced to the set of changes affecting a single row and its associated data items like index entries, text page chains etc. This means that if the transaction is rolled back mid-way, say, due to a fatal error or a constraint violation, only a few changes affecting the very last row being processed will be rolled back. All the changes done with minimal logging by the earlier commands in the transaction and changes for rows previously affected by the currently on-going statement will remain committed.

With minimal logging, essentially once the statement completes its execution, all changes remain committed to the database, and cannot be further undone by a subsequent `rollback transaction` command.

In essence, enabling ML-DML implies an acknowledgement that transactional semantics are not relevant any longer, or, alternatively, that rollbacks are never expected to occur.

Tuning ML-DML

In order to obtain gains from ML-DML, it should first be verified that minimally logged operations can actually take place: see the list of criteria governing the use of ML-DML above (and use `set show_exec_info on`). If these conditions are not met, full logging will be used after all.

Apart from this, the most important tuning aspect for ML-DML is the size of the ULC as well as the tempdb-specific ULC, defined by the configuration parameters `'user log cache size'` and `'session tempdb log cache size'`, respectively. If either of these settings are too small to contain entire subcommands, log records will be flushed to the transaction log anyway.

The default for both settings is 2048 bytes. Sybase recommends to set these configuration parameters to twice the server's page size, i.e. `@@maxpagesize*2`. Then, monitor whether this is large enough (see below), and optionally adjust the ULC sizes further (note that these ULC size settings do not have to be identical; the session tempdb ULC size could well have to be larger than the regular ULC size).

It may not be easy or even practically possible to determine the ULC size that will always be large enough to contain all log records for a subcommand. For example, for a table with long `varchar` columns and many indexes that include such columns, the cumulative size of the log records for a subcommand for inserting or deleting a row could be significant. Also, additional log records for space allocation or index maintenance (e.g. page splits) can occasionally increase the size of a subcommand.

There are two ways to determine whether subcommands fit in the ULC or not. Both require that the DML is executed first; the subcommand behaviour can then be monitored afterwards:

1. `sp_sysmon` displays counters on the subcommands:

ML-DMLs ULC Efficiency	per sec	per xact	count	% of total
Discarded Sub-Commands	33383.9	11087.8	3071323	100.0
Logged Sub-Commands	0.4	0.1	37	0.0

For a well-tuned ULC, 'Discarded Sub-Commands' should be high, indicating the majority of subcommands fit in the ULC and were therefore discarded.

2. Manually inspect the transaction log to determine if log records for subcommands appear in an IMDB or Rddb. This can be done with `dbcc log`. Look for log records of type 'SUBCMD' which are marked 'LHSC_MINIMALLY_LOGGED_DML' :

```

SUBCMD                (3068,4)  sessionid=3067,2,0
attcnt=1 rno=4 op=19 padlen=0 sessionid=3067,2,0 len=32
odc_stat=0x0020 (0x0020 (LHSC_MINIMALLY_LOGGED_DML))
loh_status: 0x20 (0x00000020 (LHSC_MINIMALLY_LOGGED_DML))
link=(3067,3) pgno=0 stat=End of sub-command,Committed

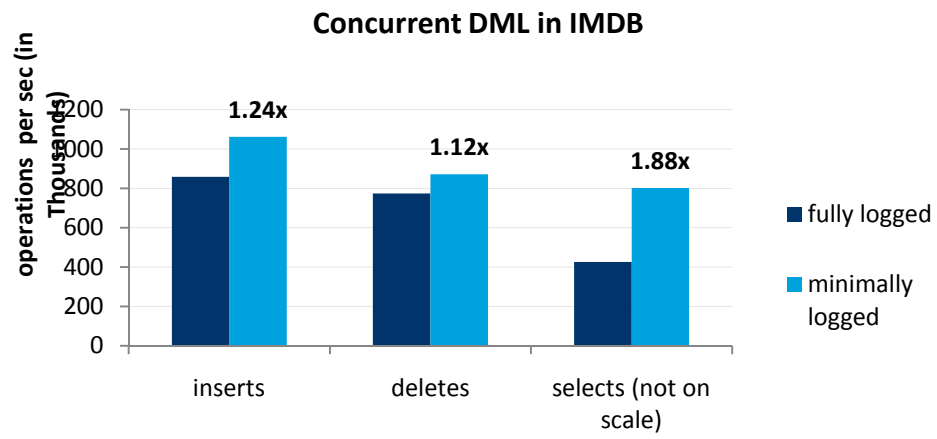
```

ML-DML : performance test results

When looking at performance aspects of ML-DML, IMDBs and RddbS should be considered separately. Note that the benefit of requiring less database space for transaction log data, which applies equally to IMDBs and RddbS, is not covered in the test results below.

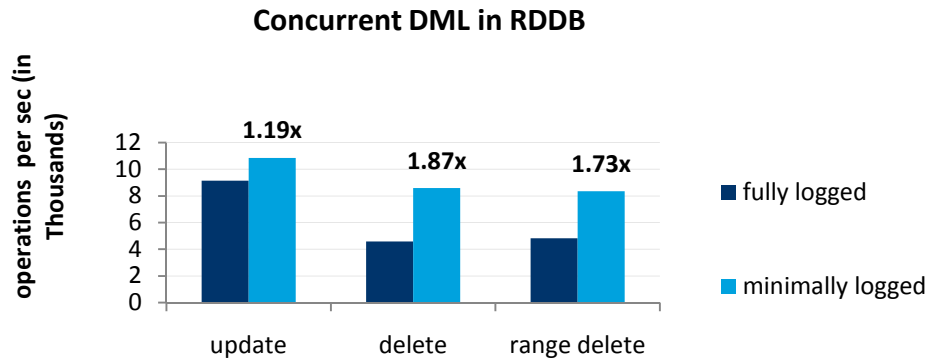
Internal performance tests at Sybase indicate that ML-DML benefits in an IMDB are minimal for an individual session with no concurrent workload. The benefits of ML-DML in an IMDB become visible in situations when many concurrent sessions are flushing their ULC to the log (mainly due to 'Full ULC'), thus creating log semaphore contention. ML-DML alleviates this contention, resulting in potential performance gains.

The test result below shows an improvement of factor 1.1 - 1.9 for different types of DML statements in an IMDB. This particular test was performed on an ASE server with 15 engines, using 45 clients concurrently performing insert, delete and select operations.



In an Rddb, disk writes will occur for log pages, as opposed to IMDBs where no disk I/O occurs at all. Consequently, Rddb's may show performance gains from enabling ML-DML for individual sessions without any concurrent workload.

The test result below shows an improvement of factor 1.2 - 1.9 for different types of single-threaded DML statements when ML-DML is enabled in an Rddb. This test was performed on an ASE server with 2 engines, using a single client session performing insert, delete and range delete operations.



These changes to the query are visible in the output of **set showplan on**, **sp_showplan**, as well as in the MDA table **monSysPlanText**.

Compatibility Mode and IMDBs/RDDBs

When accessing a table located in an IMDB, and Compatibility Mode is enabled, ASE 12.5 mode will never be used for queries involving IMDB tables.

For queries referencing tables located in an RDDB, ASE 12.5 mode can be used when Compatibility Mode is enabled.

ML-DML and the Query Optimizer

Whether or not Minimally Logged DML (ML-DML) is enabled has no impact on query plans. The decision to use ML-DML or not is made at run time and is subject to various restrictions described elsewhere in this white paper (one of those restrictions, in fact being whether the query plan uses a deferred access method or not).

Consequently, the query plan contains no information about the DML logging mode.

7. Performance Tuning: Moving from Bottleneck to Bottleneck

It is important to keep in mind that performance tuning is about addressing bottlenecks in system performance. Naturally, when one bottleneck is removed, a new bottleneck will emerge.

Tuning for IMDBs and RDBs

When using IMDBs and RDBs, classic bottlenecks around disk I/O, especially disk writes, and around transaction log handling, have been removed (see page 5).

From a performance perspective, the question should be: what will be the next bottleneck? It is not possible to predict what this will be, but here are some things that could possibly occur for IMDBs and RDBs, in no particular order. Please note that this list is in no way aiming to be complete or exhaustive.

- **Possible next bottleneck: Increased spinlock contention on ASE internal data structures.**
This can be the result from sessions more frequently accessing and modifying internal data structures now that classic reasons for short timeouts due to disk I/O have been removed. A possible example could be the spinlocks for lock manager hash table. **sp_sysmon** could provide indications of increased spinlock contention. See the ASE Performance and Tuning Guide for guidance on adjusting the corresponding '.... spinlock ratio' configuration parameters.
- **Possible next bottleneck: Increased contention on log semaphore.**
As observed in some internal test cases, increased overall throughput can have the affect of increased contention on the transaction log semaphore. This can be seen in **sp_sysmon** reports, as well as by monitoring MDA wait events (event # 54 indicates log semaphore contention). If this is observed, Minimally Logged DML could be a possible remedy.
- **Possible next bottleneck: Increased spinlock contention on IMDB inmemory_storage caches or RDBB named caches.**
Another form of increased spinlock contention might occur on the data caches associated with IMDBs or RDBs; **sp_sysmon** will clearly indicate this. If this happens, create cache partitions for the caches in question.
- **Possible next bottleneck: CPU saturation.**
Historically, many well-tuned OLTP systems have been trying to maximize the effective disk I/O bandwidth. With this aspect having become redundant for IMDBs, it could well be the case that CPU is the next resource being saturated when increasing the transaction load.
- **Possible next bottleneck: Client application algorithms.**
The client application's algorithms could become the next bottleneck. For example, if the application is not optimally designed for scalability, it may not manage to keep the ASE server fully loaded. This might not be immediately obvious, but if you see ASE spending more time waiting to receive input from the client applications (for example by monitoring the MDA wait events), and/or having a lower 'engine busy' percentage in **sp_sysmon**, then this might be happening.

Tuning for RDDBs

When using RDDBs, an important tuning aspect will be the size of the data cache used for the database. This is not fundamentally different from the considerations about cache sizing for regular user databases.

Tuning for Minimally Logged DML

When using Minimally Logged DML (ML-DML), see page 16 for information about specific tuning.

8. Using IMDBs as Temporary Databases

In ASE 15.5, a temporary database can be created as an IMDB. In this case, the various optimizations for IMDB will apply to all queries in the temporary database. Using a temporary in-memory database does not require any changes to existing queries or applications.

As an example, to create a 100MB temporary IMDB, use the following syntax:

```
create inmemory temporary database temp_imdbl
on inmem_dev = '100m'
```

Here, `inmem_dev` is an inmemory database device created on top on an `inmemory_storage` cache (see the *ASE Reference Manual* for full syntax details of these commands)

Note that temporary IMDBs cannot use a template database, since temporary databases are always recovered before user databases; and a template database is, by definition, a full-durability user database.

When using a temporary IMDB, it is recommended to experiment with enabling ML-DML for the database to see if this delivers any performance gains; this is most likely when highly concurrent workload is executed against user tables in the temporary database.

Enabling the ML-DML property can be done as follows:

```
alter database temp_imdbl
set dml_logging = minimal
```

To disable ML-DML, use:

```
alter database temp_imdbl
set dml_logging = full
```

Using RDDBs as temporary databases

Like an IMDB, an RDDB can also be used as a temporary database. The same considerations apply as for RDDBs vs IMDBs in general: an RDDB can be larger than the available memory, while it uses many of the same performance optimizations as an IMDB.

However, there are some differences in the way a temporary RDDB is created. This can be done in two ways:

a) Directly create a temporary database with an explicitly defined durability of `no_recovery`:

```
create temporary database temp_rddb1
on disk_dev = '100m'
with durability=no_recovery
```

Temporary databases always have a durability of `no_recovery`. Because of this, it is not possible to use an RDDB with `durability=at_shutdown` for a temporary database.

b) Create a regular temporary database, followed by explicitly setting its durability to `no_recovery`:

```
create temporary database my_tempdb
on disk_dev = '100m'

alter database my_tempdb set durability=no_recovery
```

Regular (non-RDDB/non-IMDB) temporary databases implicitly have a durability of `no_recovery`, which is identical to the durability setting for a temporary RDDB. However, for a temporary database to be an RDDB, and use the corresponding performance optimizations, the `no_recovery` durability must have been specified explicitly, either through `create database` or `alter database`, as shown in the examples above. Because you cannot quickly determine whether the durability has been set implicitly or explicitly, it may be difficult to distinguish these two types of temporary databases.

The quickest way to tell the difference is to enable the ML-DML attribute with `alter database ... set dml_logging=minimal`: this works for both database types, but will take effect only for a temporary RDDB. For a regular temporary database, DML logging will be done in full mode, regardless of the setting for the `dml_logging` attribute. This difference can be seen by running `set show_exec_info on` and observing the reported DML logging mode used on a simple test table.

The default `tempdb` cannot be turned into a temporary RDDB.

A temporary RDDB cannot be converted into a regular temporary database.

Using tempdb groups

Though not strictly related to IMDBs and RDDBs in the context of the Performance whitepaper, ASE 15.5 provides the possibility to create user-defined `tempdb` groups.

By creating a `tempdb` group, and binding this group to a specific login or application, that login or application will only use a temporary database from the group bound to it. This allows the DBA to manage `tempdb` resources more precisely than before. For example, it is now possible to guarantee that an application has its own (set of) temporary databases which cannot be used by anyone else.

Without user-defined `tempdb` groups, you can only bind an individual temporary database to a login or application, but not a group of temporary databases.

The example below shows how to three `tempdb` IMDBs (which are assumed to exist already) are put into a group, which is then bound to an application named 'bigapp':

```
-- Create a tempdb group:
1> sp_tempdb 'create', 'my_tmpdb_group'
2> go
```

```
-- Add user-created tempdb IMDB to the group
1> sp_tempdb 'add','temp_imdb1','my_tmpdbgroup'
2> go

-- Add another two user-created tempdb IMDBs to this group
1> sp_tempdb 'add','temp_imdb2','my_tmpdbgroup'
2> go
1> sp_tempdb 'add','temp_imdb3','my_tmpdbgroup'
2> go

-- Bind the tempdb group to the application named 'bigapp':
1> sp_tempdb 'bind','ap','bigapp','gr','my_tmpdbgroup'
2> go

-- Also bind the tempdb group to the login 'jsmith':
1> sp_tempdb 'bind','lg','jsmith','gr','my_tmpdbgroup'
2> go
```

9. Further Reading

For full syntax details about IMDBs and RDBBs, please refer to the *ASE 15.5 Reference Manual*.

This document tries to highlight some aspects of performance and tuning specifically related to IMDBs and RDBBs. For more information, see the chapter on performance and tuning in the *In-Memory Database User's Guide* which is part of the ASE 15.5 documentation set.

Obviously, many of the performance tuning principles that have always applied to ASE databases, still apply when IMDBs and RDBBs are used. Please refer to the *ASE Performance & Tuning Guide* for ample information about performance and tuning in general.

All ASE documentation mentioned above is available at www.sybase.com.

10. Summary

When using IMDBs or RDDBs in ASE 15.5, significant performance gains may be possible; improvements up to a factor 10 have been reported. IMDBs can also be used for temporary databases.

The potential for performance improvement may depend on factors such as the type of query or application, the degree of application concurrency, the number of ASE engines and the application's data access patterns.

When using Minimally Logged DML (ML-DML) in an IMDB or RDDB, apart from allowing further performance gains, less database space will need to be reserved the transaction log. This can especially be useful when performing large batch updates that are known to consume large amounts of log space.

CONTACT INFORMATION

For Europe, Middle East,
or Africa inquiries:
+(31) 34 658 2999

For Asia-Pacific inquiries:
+852 2506 8900 (Hong Kong)

For Latin America inquiries:
+770 777 3131 (Atlanta, GA)

SYBASE, INC.
WORLDWIDE HEADQUARTERS
ONE SYBASE DRIVE
DUBLIN, CA 94568-7902 USA
Tel: 1 800 8 SYBASE

www.sybase.com

Copyright © 2010 Sybase, Inc. All rights reserved. Unpublished rights reserved under U.S. copyright laws. Sybase, and the Sybase logo are trademarks of Sybase, Inc. or its subsidiaries. All other trademarks are the property of their respective owners. ® indicates registration in the United States. Specifications are subject to change without notice. 01/10.

SYBASE®