

Getting Started with In-Memory Databases in Adaptive Server® Enterprise 15.5

TABLE OF CONTENTS

1	Executive Summary
1	Introduction
2	Benefits of ASE In-Memory Databases
3	Use Cases for ASE In-Memory Databases
4	Creating & Managing In-Memory Databases
4	4 Creating inmemory_storage cache
4	4 Creating in-memory devices
5	5 Creating in-memory databases
5	5 Extending in-memory databases
5	5 Creating in-memory temporary databases
6	6 Creating & Managing RDBBs
6	6 Template databases
7	7 Minimally Logged DML (ML-DML) operations
8	8 ML-DML: Impact on transactional semantics
9	9 Dump – Load Support
9	9 Feature Offerings
9	9 License Options
10	10 Supported Platforms
10	10 Summary

EXECUTIVE SUMMARY

Sybase Adaptive Server Enterprise has proven itself for over 20 years as a highly dependable database powering mission-critical, high-performance systems in the most demanding areas of business. In version 15.5, Sybase ASE provides additional performance improvements by optionally optimizing certain aspects of database transaction handling (also known as “ACID properties”).

Many business applications contain areas where data needs to be stored only temporarily, yet ASE handles such data with the same transactional guarantees as critical data that must be stored 100% safely. By relaxing the way these transactional guarantees are enforced, ASE 15.5 now provides enhanced performance to applications which do not need to strictly adhere to ACID properties of their business transactions.

As an example, consider an e-business web application where customers shop for products. When, after a shopping dialogue, a customer checks out to complete the purchase, the corresponding data must be stored in the ASE database with full transactional guarantees, since it is not acceptable to ever lose a paid and committed order. However, the frequent and multiple updates to the shopping cart need to be very fast, but not necessarily resilient to system crashes. The data for the shopping cart needs to be stored in the ASE database, but may not require the same stringent guarantees as the final purchase: if a shopping cart is lost due to a system outage, the shopper can still start the shopping dialogue again. ASE 15.5 allows that shopping cart to be stored in an in-memory database. Since the data in an in-memory database is never stored on disk, better performance can be delivered.

Unlike in-memory products from other vendors, ASE’s in-memory database is not an additional software component that must be managed separately by the DBA. Instead, in-memory database support is fully integrated within the ASE product. Access to in-memory data is managed by ASE and can be used by applications in much the same way as any other ASE database.

ASE 15.5 provides Sybase customers with extended options to deliver best performance where possible. In all other cases, ASE continues to provide full transactional guarantees to ensure no business data is ever lost.

INTRODUCTION

Sybase Adaptive Server Enterprise (ASE) version 15.5 provides In-Memory Database (IMDB) capabilities designed to deliver low response times and high throughput to mission-critical systems.

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.

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), which is a zero-disk footprint database optimized to run completely in memory, and has no associated disk storage, and consequently, no overhead 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’ (RDDDB); this type of database is stored on disk as regular ASE databases, but has many of the same performance optimizations as IMDB databases. RDDDB 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 reduced 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 a 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.

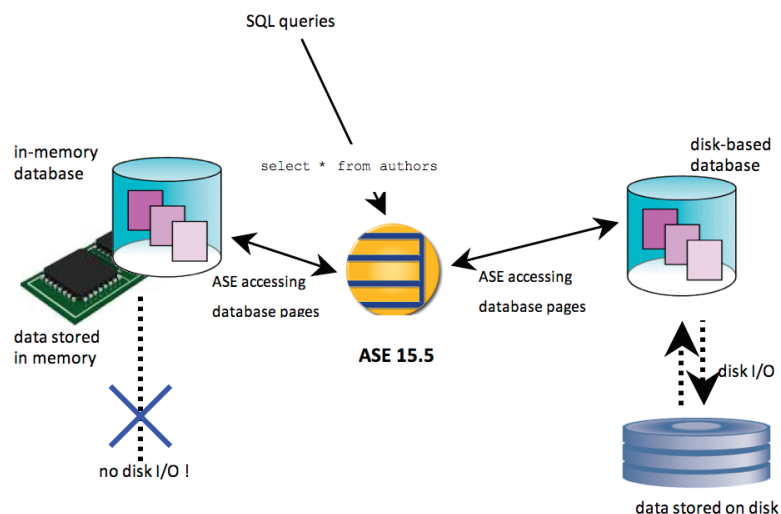
To relax the Atomic aspect of transactions (the 'A' in ACID), ASE 15.5 allows minimally logged DML to be performed in IMDB and RDDB databases.

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 guarantees, ASE 15.5 will offer customers more choices to relax ACID properties in exchange for better performance.

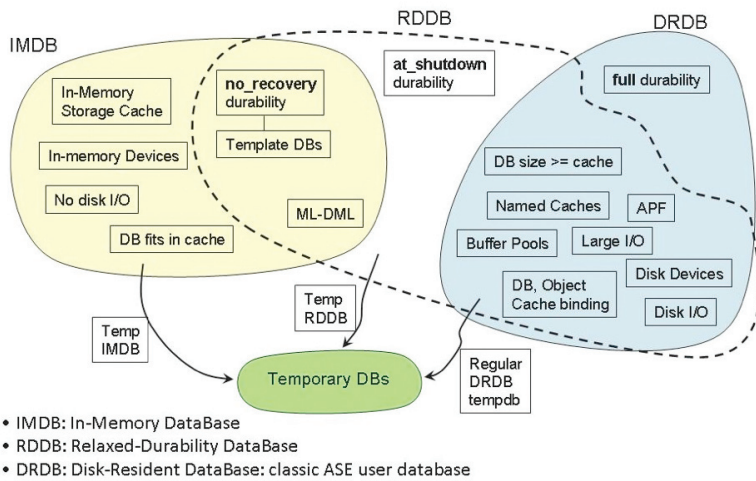
An ASE IMDB adds no complexity to the overall system architecture and operational environment. Unlike products from other vendors, ASE's in-memory database is not an additional software component that must be separately installed, configured, started and monitored. Instead, an ASE IMDB is managed by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

From the perspective of T-SQL behavior and client connectivity, IMDB databases do not have limitations in functionality: full T-SQL is supported for IMDB and RDDB databases. This includes cross-database joins, proxy tables, the ability to dump and load databases. The only difference lies in the absence of transaction durability since there is no persistent disk storage. This is illustrated by the following diagram:



The ASE query optimizer takes the 'zero-disk' aspect of IMDB database into account when generating query plans.

The following diagram shows properties of different flavors of databases supported in ASE:



Use Cases for ASE In-Memory Databases

ASE In-Memory databases with relaxed ACID properties can be useful in many situations, for example:

- 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 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).

Also, 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.

CREATING & MANAGING IN-MEMORY DATABASES

One should create the resources required for the in-memory database before creating the database. These resources include the actual storage cache and in-memory devices created from the cache.

Creating `inmemory_storage` cache

`inmemory_storage` cache is no different from regular named caches. `sp_cacheconfig` with the new cache type `inmemory_storage` will create the named cache for hosting the in-memory database with `NONE` as the buffer replacement strategy. The following example creates the named cache `imdb_cache` for the in-memory database:

```
1> sp_cacheconfig 'imdb_cache', '300M', 'inmemory_storage'  
2> go
```

The change is completed. The option is dynamic and ASE need not be rebooted for the change to take effect.

For `inmemory_storage` caches, `sp_helpcache` reports the different in-memory devices created in the cache, the memory utilization by these in-memory devices, and free space available.

Creating in-memory devices

An in-memory database can be created directly on an `inmemory_storage` cache with just one additional step. Analogous to disk devices, the DBA must first create in-memory database devices for creating the database.

The interface for creating in-memory devices is `disk init` with the new option `'type=inmemory'` and `'physname=<cache_name>'`. An in-memory device is a virtual cache device created on top of the `inmemory_storage` cache i.e. each device represents a chunk of cache memory.

The purpose of an in-memory device is to support the device level space tracking features like segments and threshold management for the in-memory databases, and for supporting databases with dedicated log segments

The following examples create two devices on the `imdb_cache`:

```
1> disk init name = 'imdb_datadev1', physname = 'imdb_cache',  
2> size = '250M', type = 'inmemory'  
3> go
```

```
1> disk init name = 'imdb_logdev1', physname = 'imdb_cache',  
2> size = '50M', type = 'inmemory'  
3> go
```

`sp_helpdevice` has been enhanced to display the in-memory device information.

Creating in-memory databases

In-memory databases are created using the `create database` command, referring to the in-memory devices created earlier. The `create database` command establishes implicit one-one cache binding between the new database and cache (i.e. an in-memory storage cache can host only one in-memory database and the in-memory database cannot be created on multiple caches). In-memory databases must be created with durability option `no_recovery` durability (i.e. these databases will be re-created upon server boot).

The following command will create an in-memory database named `imdb` on the in-memory devices created in the previous section:

```
1> create inmemory database imdb
2> on imdb_datadev1 = '250M'
3> log on imdb_logdev1 = '50M'
4> with durability = no_recovery
5> go
Warning: Database 'imdb' has a durability level of NO_RECOVERY; changes to it will be lost when you restart Adaptive Server.
Database 'imdb' is now online.
```

`sp_helpdb` has been enhanced to display the in-memory database information such as its durability and the cache in which the database has been created.

Extending in-memory databases

If unused space exists on the in-memory devices in the `inmemory_storage` cache or some unused in-memory devices exist on the cache on which the database's device(s) reside, the database can be extended to use this unused space. Extending the size of an in-memory database is similar to adding more disk space using `alter database` to a regular database. However, the procedure to be followed is slightly different:

1. Alter the database to first use up space on in-memory devices previously created from the cache and then from unused devices in the same cache.
2. Extend the size of the `inmemory_storage` cache using `sp_cacheconfig`
3. Create a new in-memory device on the extended cache using `disk init`
4. Extend the in-memory database on the newly created device using `alter database`

Creating in-memory temporary databases

Many queries and applications use `tempdb` for worktables and for temporary tables that hold intermediate results.

ASE 15.5 adds in-memory temporary databases to help improve application performance by utilizing optimizations specific for in-memory databases. In-memory temporary database are functionally identical to regular temporary databases.

The following example creates an in-memory temporary database :

```
1> create inmemory temporary database temp_imdb
2> on timdb_dev1 = '100M'
3> with durability = no_recovery
4> go
Database 'temp_imdb' is now online.
```

ASE 15.5 supports user-created temporary database groups, some of which can now be created only from in-memory temporary databases, and others defined to contain disk-resident temporary databases.

CREATING & MANAGING RDDBS

A Relaxed-Durability Database (Rddb) can be used in situations where the database size is too large to fit entirely into the available memory, so an IMDB cannot be used.

Creating a Relaxed Durability Database (Rddb) is almost identical to creating a traditional disk-based user database. One should first create regular disk-based database devices (using `disk init`), followed by `create database`, to create an Rddb. An Rddb inherits many of the same optimizations for in-memory databases and provides two types of durability semantics.

- `no_recovery` durability database is identical to an IMDB (i.e. at server re-start, the Rddb can be re-created from the template database).
- `at_shutdown` durability database guarantees the consistency of the database after a polite shutdown command (i.e. these databases can be recovered cleanly after a polite shutdown). On the other hand, if the server is stopped by `shutdown with nowait`, the boot recovery process will not be able to recover the database. The database will be marked suspect and will be kept in an off-line state.

Creating Relaxed Durability Databases

Relaxed durability databases are created on disk devices like other traditional databases. One should specify the durability level along with the device list in the `create database` statement. The following example will create an RDDBs with `no_recovery` durability:

```
1> create database rddb
2> on rddb_dev1 = '400M'
3> with durability = no_recovery
4> go
Warning: Database 'rddb' has a durability level of NO_RECOVERY; changes to it will
be lost when you restart Adaptive Server.
Database 'rddb' is now online.
```

To create an Rddb with `at_shutdown` durability, use the syntax `with durability = at_shutdown` instead.

Note that omitting the clause `'with durability=...'` will create a regular disk-based database with full durability.

TEMPLATE DATABASES

By default, ASE creates new databases by using the 'model' database as a blueprint, and copying its contents into the newly created database.

ASE 15.5 provides the option of specifying any other disk-resident user database with full durability as the template database for creating IMDBs and RDDBs with `no_recovery` durability.

When a template database is used, the newly created IMDB or Rddb will inherit all properties (including database options like `'select into/bulk copy/p11sort'`) and objects (i.e. tables, stored procedures, views etc.) of the template database. For an IMDB, if there are any cache bindings associated with the template database, these will not be inherited by the in-memory database (because an IMDB cannot be bound to multiple named caches).

The template database feature is supported for non-recoverable databases only, because they need to be re-created as a part of the boot recovery process. If these databases are created with the 'model' database, then customers may need to load a database dump, or run a SQL script, every time the database is recreated. Using template databases avoids these repetitive tasks, thereby greatly reducing the application startup time.

`sp_helpdb` has been enhanced to report the relationship between the non-recoverable database and user-defined template database. When a template database is used for creating non-recoverable databases, the template database:

- cannot be dropped till all dependent databases created from it are first dropped;
- cannot be altered to increase the database size beyond the size of the smallest database created from it (i.e. any number of in-memory databases or `no_recovery` RDDBs can be created from a single template database).

The following examples will create an IMDB specifying a template database:

```
1> create inmemory database imdb
2> use small_db as template
3> on imdb_datadev1='250M'
4> log on imdb_logdev1='50M'
5> with durability = no_recovery
6> go
Warning: Database 'imdb' has a durability level of NO_RECOVERY; changes to it will
be lost when you restart Adaptive Server.
Database 'imdb' is now online.
```

Users can create IMDBs and non-recoverable RDDBs from the same template database. If there are multiple databases created from a single template then they will be re-created serially.

MINIMALLY LOGGED DML (ML-DML) OPERATIONS

Using ASE 15.5, applications can now perform large batch operations, such as multi-row updates or deletes, resulting in enhanced performance by using minimal logging. Transaction logging is performed in the User Log Cache solely to support run-time rollback, and with a properly configured ULC size, no log records are flushed to the transaction log. In workloads with concurrent and heavy DML activity, the bottlenecks created while trying to persist the changes to the transaction log can be substantially reduced, thereby significantly improving the performance of concurrent DML when executed with minimal logging.

Minimal logging for DML is supported on all types of low-durability databases, such as IMDBs and RDDBs. If the user tries to configure minimally logged DML (ML-DML) in a database with full durability, it will be silently ignored and full logging will be used. Users can configure minimal DML logging per database, per table and on session level. The ML-DML feature will be effective only when the '`select into/bulkcopy/p11sort`' database option is enabled.

Minimal DML logging can be configured at the level of a database, table or session:

1. **Database Level DML Logging:** If minimal logging is enabled at the database level, all the DML operations on the user and temporary tables may not log all the changes in the database. The minimal logging feature is not applicable to system databases including the 'model' database. However, one can set ML-DML for the objects in the 'model' database. These objects will then be copied along with their minimal logging setting into newly created low-durability databases. Temporary databases are major candidates for the database level minimal logging feature, because:
 - it improves the application performance by reducing the number of log I/Os and
 - there is no need to change the application except for changing the database level logging option.

The following examples create databases with the minimal logging option.

```
1> create inmemory database imdb on imdb_datadev1 = '250M'
2> log on imdb_logdev1 = '50M'
3> with durability = no_recovery, dml_logging = minimal
4> go
Warning: Database 'imdb_db_minlog' has a durability level of NO_RECOVERY; changes
to it will be lost when you restart Adaptive Server.
Database 'imdb_db_minlog' is now online.
```

The `dml_logging` property of a database can be changed between `minimal` and `full` dynamically using the `alter database` command.

- 2. Table-Level DML Logging:** Minimal DML logging mode can also be set at table level. This will enable users to configure a set of tables (such as the most frequently accessed tables) to use minimal DML logging. The table level DML logging option will override the database level setting.

The following example shows how to creating a tables with the `dml_logging` option enabled in an RDDB or IMDB:

```
1> use rddb
2> go

1> create table mldml_tab(col1 int, col2 char(100))
2> with dml_logging = minimal
3> go
```

The `dml_logging` property of a table can be changed dynamically between `minimal` and `full` using the `alter table` command.

- 3. Session-Level DML Logging:** Users can configure minimal logging for all DML executed in a session. This setting will override the table level and database level settings. It is applicable only to tables owned by the session's user. If the session user has `sa_role`, then the logging mode of all user objects will be affected by the session level logging level. Session-level minimal DML logging is enabled by using this command:

```
set dml_logging { minimal | default }
```

After the user sets session-specific DML logging to `minimal`, running `set dml_logging default` returns the logging mode currently in effect for the affected tables to the tables' default logging mode, based on the table and database level logging settings.

ML-DML: Impact on transactional semantics

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

DUMP – LOAD SUPPORT

ASE 15.5 provides fully integrated support with Backup Server to dump and load IMDBs or RDDBs.

You can dump an in-memory database directly from ASE shared memory to a disk-resident archive, and then load it back directly from the on-disk archive to ASE shared memory in an in-memory database. Dumps of in-memory databases can be loaded in other disk-resident databases as well. Full support and recovery of database dumps for RDDBs is also supported. Likewise, loading dumps of databases with full durability into an in-memory database, or to other RDDBs is also supported.

Complete recoverability of ongoing transactions in IMDBs or RDDBs is provided by the ASE recovery manager, even with the use of minimal logged DMLs, or with relaxed transaction durability. This allows for periodically persisting in-memory data to disk-resident storage. The archive process is fully integrated with existing Backup Server technologies such as backup compression, and with Tivoli Storage Manager.

Seamless integration of the archive techniques for IMDBs and RDDBs with existing Backup Server technology greatly simplifies assimilating these new database offerings into your existing information management infrastructure.

FEATURE OFFERINGS

The following tables give the set of all other database level commands and stored procedures modified in ASE 15.5 to support the new features like IMDB, RDDB and Minimally Logged DML.

Command	Description
dump database	Dumps Sybase ASE In-Memory Database data to an archive device.
load database	Loads Sybase ASE database archive into in-memory database.
alter database	Supports extending the size of the database, change template database, and modify the DML logging and durability levels.
drop database	Drops the given database.

Stored Procedure	Description
sp_helpdevice	Reports the information pertaining to virtual cache or inmemory devices.
sp_helpdb	Modified to print the information about the inmemory devices, template, durability and dml_logging information.
sp_helpcache	Modified to print the in-memory device information along with the status (i.e. actively used by device or free space)
sp_addsegment	Modified to define a segment on the in-memory device
sp_helpsegment	Reports the segment information associated with the in-memory device.
sp_help	Reports DML Logging information associated with the table.

LICENSE OPTIONS

The Sybase ASE In-Memory Database feature is a Sybase licensed option. This option is available in Sybase ASE 15.5 Enterprise Edition and Developer Edition.

Note that the In-Memory Database feature is currently not supported for ASE Cluster Edition.

Customers need to purchase an 'ASE_IMDB' license of Sybase ASE 15.5 to enable the IMDB and RDDB functionality. Please refer to the Sybase ASE documentation for further details on licensing.

SUPPORTED PLATFORMS

Sybase ASE 15.5 supports IMDB, Rddb and ML-DML features on the following platforms:

- HP-UX Itanium 64-bit
- HP-UX PA RISC 64-bit
- IBM AIX 64-bit
- Linux on Power 64-bit
- Linux Opteron 64-bit
- Solaris Sparc 64-bit
- Solaris Opteron 64-bit
- Windows Opteron 64-bit

SUMMARY

ASE 15.5 provides new in-memory database (IMDB) capabilities that offer customers additional options to achieve better performing applications. These performance improvements are realized by optionally relaxing the 'Atomic' and 'Durable' properties of ACID transaction behavior. By relaxing these aspects, ASE 15.5 can avoid disk I/O altogether, as well as optimize the maintenance of the transaction log; thereby delivering improved performance. While relaxing these aspects may not be suitable for all applications, there are many areas in business applications where this would be acceptable, in return for better performance.

An ASE IMDB 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 managed separately by the DBA. Instead, an ASE IMDB is managed by ASE and can be used by applications just as any other ASE database. As an ASE-managed database, IMDB provides full T-SQL capabilities and client connectivity. Apart from bringing performance benefits, ASE IMDBs do not require any disk storage as they run completely in memory. For databases that are too large to fit in memory, ASE 15.5 provides Relaxed Durability Databases (Rddb) which are stored on disk but provide many of the same optimizations as IMDBs.

ASE 15.5 delivers value to Sybase customers by providing improved performance and reduced disk storage requirements.