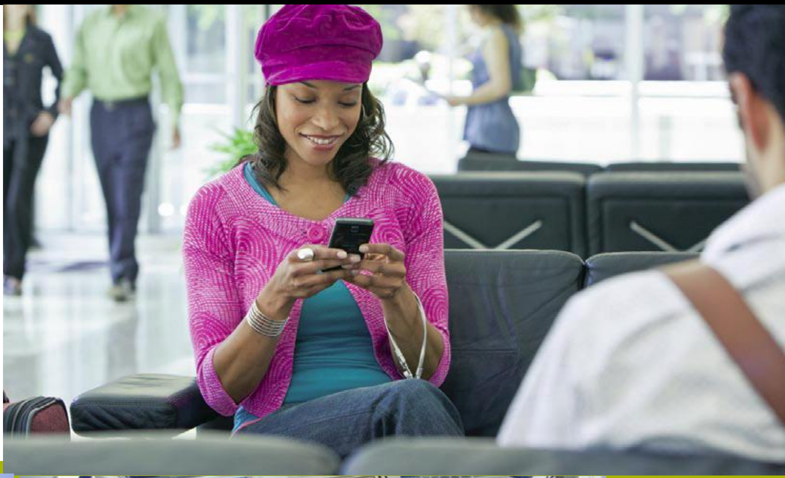
SYBASE
An SAP Company

Using Replication with Sybase ASE In-Memory Databases

TECHNICAL WHITE PAPER



www.sybase.com

Contents

Revision History

Version 1.0 - August 2011

Version 1.1 - June 2012: various edit and corrections

Executive Summary

Version 15.5 of Sybase Adaptive Server Enterprise (ASE) introduced the In-Memory Database (IMDB) feature which aims to provide improved performance by relaxing transaction durability properties. As a consequence, these databases do not provide persistence over an ASE restart. In turn, this means that such IMDB databases require special setup procedures to allow their use in combination with Sybase Replication Server.

Using IMDB databases, both as source and target of replication, is supported with the combination of ASE 15.5 ESD#4 and Replication Server 15.6 ESD#1. This whitepaper describes how to set up transactional replication with ASE IMDB databases.

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.

In most ways, an ASE IMDB is no different from any other database managed by ASE. 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 databases. This includes cross-database joins, proxy tables, and the ability to dump and load databases. The only difference lies in the absence of transaction durability since there is no persistent disk storage in an IMDB.

However, this absence of persistent disk storage represents a complication when using an IMDB database for replication purposes. This is because Sybase Replication Server expects the following information in a primary or replicate database to be stored persistently:

- The secondary truncation point (in a primary database)
- The configuration of the Replication Agent (in a primary database)
- Primary tables being marked for replication
- The maintenance user (in a replicate database)
- The ID of the most recently committed transaction in the `rs_lastcommit` table (in a replicate database)

Supported ASE & RS versions

Some internal adaptations have been made to both ASE and Replication Server in order to use a primary IMDB or RDDB with `durability=no_recovery`, as a replication source.

This is supported with the following product versions (both are required):

- ASE version 15.5 ESD#4 (or later)
- Replication Server version 15.6 ESD#1 (or later)

It should be noted that earlier versions of ASE and Replication Server already implemented some elements related to IMDB replication. However, for technical reasons the above mentioned versions are both required for replicating from an IMDB in a supported manner.

Terminology: IMDB vs. RDDB

In the context of this paper, 'IMDB' stands for 'In-Memory Database', which is an ASE database running entirely in cache.

'RDDB' stands for 'Reduced Durability Database', which is a disk-based ASE database with many of the same enhancements related to transaction logging as in an IMDB. Two types of RDDBs exist: one `with durability=no_recovery`, which is re-initialised from scratch after every ASE restart; and one `with` `durability=at_shutdown` which provides transaction durability only in case the ASE server was shutdown in a normal manner.

In this paper, 'IMDB' and 'IMDB/RDDB' should be read as 'an IMDB or RDDB `with durability=no_recovery`'. For the purpose of replication, these two database types behave identically. In some cases, the text will explicitly mention both types, but for brevity this is not always the case.

In contrast, an RDDB database `with durability=at_shutdown` does provide persistent storage over ASE restarts, and therefore behaves the same as any other regular database as far as replication is concerned. This database type is therefore not discussed in this paper.

Note that temporary databases also have `durability=no_recovery`; however, since temporary databases cannot have a template database, the replication setup described in this paper does not apply.

A formal view on IMDB replication

Without the product adaptations and setup procedures described in this paper, formally speaking, replication from an IMDB is still possible.

Indeed, replication works normally for IMDB/RDDB databases (both primary and replicate) as long as the ASE server is not restarted: once the IMDB database connection and replication definitions have been created, replication will keep working fine until the next ASE shutdown.

Because of the lack of persistent storage over ASE restarts in IMDB/RDDB, the DBA would have to recreate the entire replication setup for such databases after every ASE restart to ensure replication keeps working correctly. This would mean dropping and recreating the IMDB database connection, and the corresponding replication definitions and subscriptions. While there is no reason why this would not work, few customers would likely consider this a workable situation.

Since ASE restarts should be expected to occur, the above replication setup would be a rather academic case with little practical applicability.

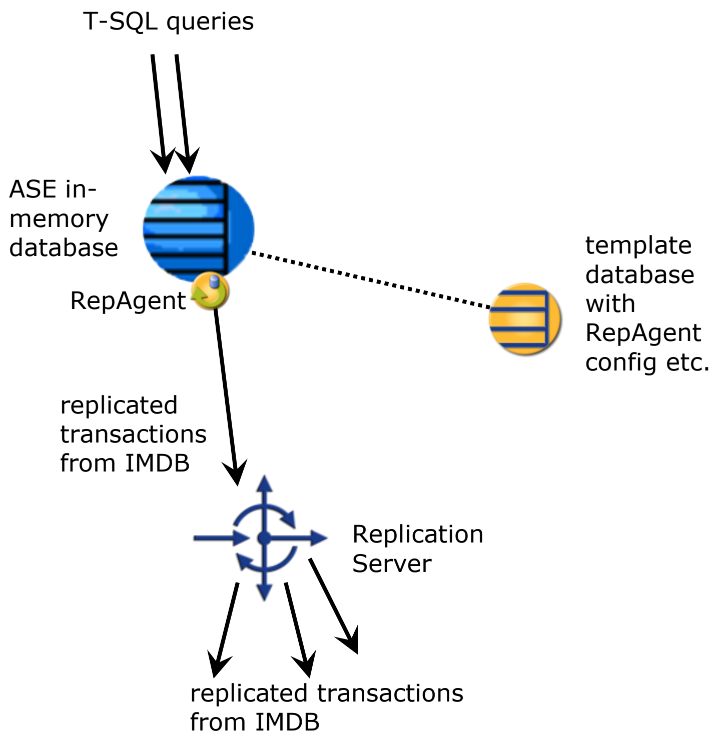
The focus of this paper is on the more realistic scenario whereby ASE restarts happen every now and then, and where replication must keep working without the need to recreate database connections, replication definitions (etc.) every time.

Single primary IMDB/RDDB based on a template database

When using an IMDB or RDDB database with `durability=no_recovery` as a primary database for replication, some special setup steps are required to make things work as you'd expect.

The main difference with a regular, full-durability primary database is that the IMDB/RDDB database `with durability=no_recovery` must be set up with a template database. This template database contains the tables, stored procedures and replication status of all objects, the secondary truncation point, the ASE Replication Agent configuration, the `rs_lastcommit` table, and the maintenance user.

When ASE restarts, the IMDB/RDDB database is automatically recreated by copying the template database, causing the IMDB/RDDB to inherit the full contents and replication-related settings of the template database.


T-SQL queries
ASE in-memory database
RepAgent
template database with RepAgent config etc.
replicated transactions from IMDB
Replication Server
replicated transactions from IMDB

Replicated primary IMDB with template database

The following setup steps are required (a full code example is included in the appendix, see page 18):

1. First create a full-durability database with the same name as the final IMDB. Let's assume our final primary IMDB will be named **'prim_imdb'**, so create a full-durability database with this same name:

```
create database prim_imdb
on my_disk_device = '50M'    -- template may be small
with durability=full         -- default, may be omitted
```

2. Next, use **rs_init** to add this database to the replication system, marking it as a replicated primary database. This will configure the secondary truncation point, ASE Replication Agent and maintenance user in **prim_imdb**.

   NB: it is recommended to create an **rs_init** resource file by hitting **CTRL+W** after completing the **rs_init** dialog but before executing the requested actions. This will be especially useful when setting up multiple IMDBs/RDDBs with the same template, as described on page 10. With a resource file, **rs_init** can be run from the command line without an interactive dialog.

At this stage, the DBA should verify that replication from **prim_imdb** actually works (i.e. create a dummy table, mark it for replication, create a replication definition and subscription and replicate some transactions). If replication works fine, proceed with the next steps:

3. Stop the ASE Replication Agent in **prim_imdb**, and (in Replication Server) suspend the connection to **prim_imdb**. Then rename **prim_imdb** (with **sp_renamedb**), for example to **imdb_template** (this requires putting the database in single-user mode first with **sp_dboption**; remove this again after renaming)
4. Create the actual primary IMDB database named **prim_imdb**, with **imdb_template** as its template:

```
create inmemory database prim_imdb
use imdb_template as template
on imdb_device = '1G'
with durability = no_recovery
```

5. Now start the ASE Replication Agent in **prim_imdb**, and in Replication Server, resume the connection to **prim_imdb**, and verify that replication still works by performing some transactions on a replicated table in **prim_imdb**. From the perspective of Replication Server, nothing has changed about the primary database (except for the secondary truncation point, which is discussed below).

When ASE is shut down and restarted, the **prim_imdb** database is recreated from its template. This puts the primary IMDB in the same state as after step 4 above. Once the ASE Replication Agent is started, replication can continue.

6. (Optional) When **prim_imdb** also acts as a replicate database, the maintenance user (**prim_imdb_maint**, by default) may need to be aliased to the **dbo** user (this is often done to work around permission problems and to avoid having to assign **sa_role** or **replication_role** to the maintenance user login). This requires dropping the maintenance user from the template database and adding it again as a **dbo** alias. See page 21 for an actual code example.

Notes:

- As an IMDB/RDDB with `durability=no_recovery` does not provide persistent storage, transactions may be lost if ASE is shut down in the middle of processing: any transactions in the primary IMDB that have not yet been read and sent by the ASE Replication Agent, are lost.

- It is not supported to configure the template database itself as a real primary database (i.e. where actual transactions are replicated from).
- Any changes to tables in the primary IMDB/RDDB (schema, replication status), should always be made identically in the template database.
- Any changes to the ASE Replication Agent configuration in the primary IMDB/RDDB should also be applied to the ASE Replication Agent configuration in the template database.
- When making any changes or updates in the template DB, always truncate the transaction log of the template DB. When this is avoided, such changes or updates in the template DB can lead to log records in the transaction log of the template DB, which are copied into the IMDB/RDDB after an ASE restart; if these log records are for replicated tables or procedures, they will be picked up by the ASE Replication Agent and sent to Replication Server. This scenario should probably be avoided, so truncate the log of the template DB after making any changes.

Inheriting database-level settings from the template database

Various database-level properties can be set. Depending on the setting, the property may or may not be automatically inherited from the template database:

- When the template database is marked for replication with **sp_reptostandby**, this setting is inherited by the IMDBs/RDDBs configured with this template. The same applies for settings made through with **sp_setrepdbmode**. This is because the settings by **sp_reptostandby** and **sp_setrepdbmode** are stored inside the database itself, and are therefore copied into the final IMDB/RDDB.
  (incidentally, for the same reason the Replication Agent configuration is inherited from the template database).
- Database settings made through **sp_dboption**, such as **'trunc log on chkpt'** or **'select into/bulkcopy/pllsort'**, are inherited from the template database. These settings are persistent over an ASE restart (even though the IMDB/RDDB itself is not).

Automatically starting the IMDB Replication Agent

In order to let the ASE Replication Agent in the primary IMDB start automatically after an ASE restart, configure the ASE Replication Agent in the template database as 'started'. The effect is that when ASE restarts and the IMDB/RDDB is recreated from this template database, the ASE Replication Agent in the IMDB/RDDB will then start automatically.

Depending on the ASE version this should be configured as follows:

- In ASE 15.5 ESD#5 or later, run **sp_config_rep_agent imdb_template, 'auto start', true**. This will not start the RepAgent in the template DB itself, but will cause the RepAgent in the IMDB to be started automatically after ASE restart.
- In ASE versions earlier than 15.5 ESD#5, run **sp_start_rep_agent imdb_template**. This will cause the RepAgent in the IMDB to be started automatically after ASE restart.

  As a side effect, it also attempts to start the RepAgent in the template DB. This will fail since the template DB is not configured as a Replication Server connection. As a result, an error message will occur in both the ASE and Replication Server errorlog files. This error message is harmless and can be ignored for the template DB.

Note that when the IMDB/RDDB is manually created with a **create** [**inmemory**] **database** statement, this does not start the ASE Replication Agent automatically; this only happens at ASE startup time.

If the ASE Replication Agent in the template database is not marked as started, the ASE Replication Agent in the IMDB will remain stopped after an ASE restart and must be started manually.

Initializing an IMDB/RDDB with dump/load

For completeness, instead of using a template database for a primary IMDB/RDDB, it is also possible to use dump and load for initialization. This would require:

1. Create a replicated database with primary tables, ASE Replication Agent configuration, etc.
2. Making a database dump of this primary database
3. After ASE restart, loading this database dump in every primary IMDB/RDDB, and onlining the database.

However, method is less attractive than using a template database because:

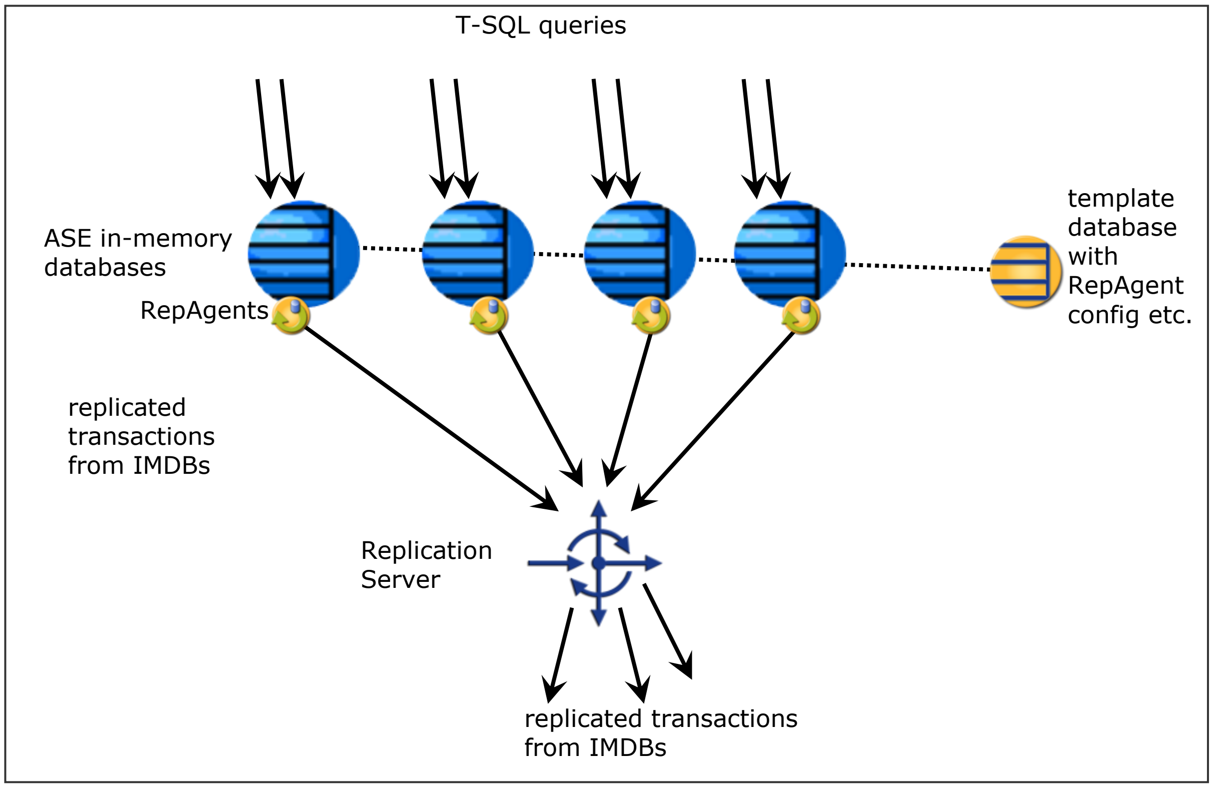
- Database dumps are not automatically loaded after an ASE restart, meaning that the DBA has to take specific action in order to prepare the primary IMDB/RDDB as a replication source or target. In contrast, when using a template database, the IMDB/RDDB are automatically copied from the template, and are immediately ready for use after an ASE restart, without any DBA intervention.
- After loading a dump and onlining a database which will act as a replication primary, the ASE Replication Agent is not automatically started, so this requires DBA intervention. When using a template database, the ASE Replication Agent can be started automatically after an ASE restart.

Due to these limitations, this paper will not discuss the dump/load approach, but instead focus on using a template database.

Multiple primary IMDBs/RDDBs with a shared template

In the previous section, the situation described was that of a single primary IMDB/RDDB with its own template database.

However, it may well be desired to have multiple identical primary IMDBs/RDDBs with identical schemas in an ASE server, for example in scale-out scenarios where an incoming workload is spread over multiple IMDB databases. While it is certainly possible to create a separate template database for each such primary IMDB/RDDB, it may be more convenient to use a single template database for multiple identical primary IMDBs/RDDBs.


T-SQL queries
ASE in-memory databases
RepAgents
template database with RepAgent config etc.
replicated transactions from IMDBs
Replication Server
replicated transactions from IMDBs

Multiple replicated primary IMDBs with a single template database

For setting up multiple identical primary IMDBs/RDDBs with the same template database, follow the setup steps below (see page 21 for a full code example). Note that this requires that all IMDBs/RDDBs connect to the same Replication Server.

1. First create the template database and the first primary IMDB/RDDB as described on page 6.

For each additional IMDB/RDDB that must also be a replicated primary database based on the existing template database, follow these steps:

2. Create the database with the desired name of the primary IMDB databases, for example **prim_imdb2**. Important: do not create the database with a template at this point. In fact, it is sufficient to create this as a simple and small disk-based database at this point:

   ```
   create database prim_imdb2
   on my_disk_device = '10m'
   ```

3. Next, use **rs_init** to add this database to the replication system, marking it as a replicated primary database. If an **rs_init** resource file was created as suggested on page 7, it is sufficient to edit the resource file and change the name of primary database, and then run:

   rs_init –r *<resource-file-name>*

4. Stop the ASE Replication Agent in **prim_imdb2**, and then drop database **prim_imdb2**.
5. Recreate database **prim_imdb2** as an actual IMDB (or RDDB), this time using the template database **imdb_template**.
6. Start the ASE Replication Agent on **prim_imdb2**.

At this point, replication from **prim_imdb2** is ready to start once replication definitions for primary data in this database, as well as subscriptions, have been created.

Steps 2-6 can be repeated for additional primary IMDBs/RDDBs.

Using an IMDB/RDDB as a replicate database

Using an IMDB/RDDB database `with durability=no_recovery` as a replication target is supported since ASE 15.5, with any version of Replication Server. This requires a similar setup with a template database as when using an IMDB/RDDB as a primary database.

The issue with using an IMDB/RDDB as a replicate database is preservation of the following data over an ASE restart:

- the maintenance user (used by Replication Server to apply replicated transactions)
- the `rs_lastcommit` table

These two elements must be present in a replicate database for Replication Server to connect successfully. The solution is similar as when using an IMDB/RDDB as a primary database, namely to use a template database that contains the maintenance user as well as `rs_lastcommit`.

The setup procedure is therefore similar to the steps described on pages 6-7, except that configuring the replication agent and marking primary data for replication is not required in the template database.

Multiple replicate IMDBs/RDDBs with a shared template database

When setting multiple IMDBs/RDDBs as replicate databases based on the same template database, a similar procedure should be followed as when setting up multiple primary IMDBs with a single template, although no Replication Agent configuration is necessary.

However, an additional setup step is required to configure the correct maintenance user for every IMDB/RDDB in the template. The maintenance user must be explicitly added to the template database, optionally as an alias of the **dbo** user, if so desired. For an actual code example, see page 23.

Using autocorrection for replicate IMDBs/RDDBs

When an IMDB/RDDB is used as a replicate database, the ID of the last committed transaction in `rs_lastcommit` is lost after an ASE restart. This can lead to a situation where Replication Server re-applies undeleted transactions from the outbound queue to the replicate database after an ASE restart.

In some scenarios, for example when data is replicated to both the template and the replicate IMDB itself, this could lead to duplicate key errors (or to other errors as Replication Server re-applies undeleted transactions from the outbound queue). If such a risk exists, autocorrection should be enabled in Replication Server for the relevant table replication definitions.

Replicating both to the template and its replicate IMDBs/RDDBs

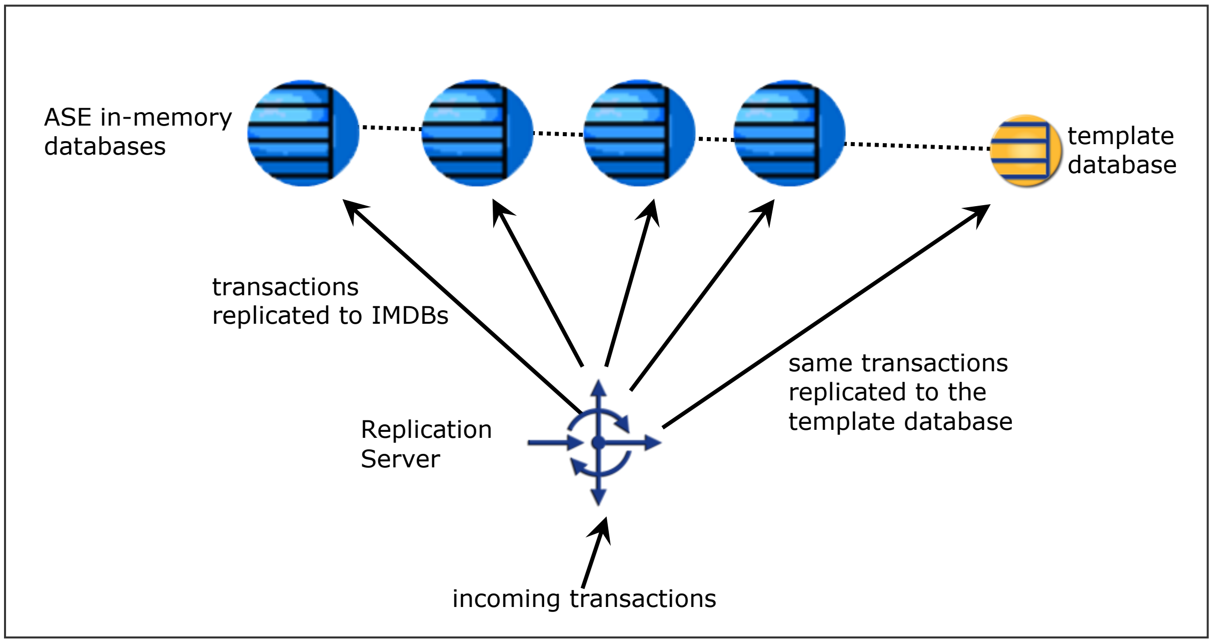
In scenarios where a replicate IMDB/RDDB database cannot afford to lose any transactions, a possible architecture is to replicate the same data to the IMDB/RDDB as well as to the corresponding template database at the same time.

Let's assume data is replicated directly to the replicate IMDB/RDDB to keep that database up-to-date with an incoming data stream. However, when ASE restarts, all replicated transactions that were applied to the IMDB/RDDB, are lost. A solution can be replicating the same data also the template database, thus keeping it also up-to-date. As a result, the IMDBs/RDDBs that use this template will immediately be up-to-date as well after an ASE restart.

However, please note the following.

Since, after an ASE restart, the template database is brought online before the IMDBs/RDDBs are recreated, Replication Server can start applying transactions to the template database. This could lead to a situation where the IMDBs/RDDBs are created with different contents, since they were based on different versions of the template database (since it is being changed by Replication Server). To avoid this, suspend the DSI to the template database until the IMDBs/RDDBs are recreated.

In addition, it may be needed to use autocorrection for the replicate IMDBs/RDDBs after an ASE restart since Replication Server might attempt to apply previously replication transactions again to the IMDB/RDDB.


ASE in-memory databases
template database
transactions replicated to IMDBs
same transactions replicated to the template database
Replication Server
incoming transactions

Replicating transaction both to the template DB and to replicate IMDBs

Using an IMDB/RDDB both as primary and replicate database

When setting up a database as a replicated primary database in Replication Server (as described on pages 6-7), that same database can automatically be replicate destination as well. In this sense, IMDBs/RDDBs are no exception to regular databases being used for replication.

Supported types of replication

Once an IMDB or RDDB database has been set up as a primary and/or replicate databases, such a database can participate in all types of replication, such as:

- table replication (with table replication definitions)
- replicating stored procedures (with function replication definitions)
- database replication/MSA (with database replication definitions)
- replication of text/image data
- warm standby (with a logical connection)
- bidirectional replication

While an IMDB/RDDB is just another database from the perspective of Replication Server, whether or not all these forms of replication are best suited to work with an IMDB or RDDB is another matter. It is therefore recommended to always consider the implications of using an IMDB or RDDB in a replication system, and pay special attention to:

- the possibility that primary transactions in an IMDB/RDDB are lost because they have not yet been picked up by the Replication Agent at the moment when ASE shuts down.
- the possibility that, after an ASE restart, transactions destined for a replicate IMDB/RDDB are still in the Replication Server queues despite having already been applied before the ASE shutdown. Due to `rs_lastcommit` contents not being persistent in the replicate IMDB/RDDB, such transactions will be applied again to the IMDB/RDDB, which may or may not be desirable.
- recovery scenarios whereby replication needs to be restored by loading earlier dumps/backups.

Duplicate Detection and replication from an IMDB

Setting up replication for a primary IMDB/RDDB database `with durability=no_recovery` with a template database containing the replication configuration, as described in this paper, is quite straightforward.

However, experienced Replication Server users will have spotted a problem, namely "duplicate detection". Indeed, to make replication from a primary IMDB/RDDB work, ASE and Replication Server have been enhanced to automatically disable duplicate detection for that primary IMDB/RDDB database, as described below.

Duplicate Detection: some background

A central concept in Replication Server is the unique ID identifying each replicated transaction, the so-called **OQID** (also known as 'locator'). This is a 36-byte binary value that is attached to each replicated transaction by the ASE Replication Agent in the primary database.

By design, the OQID is a universally increasing quantity, meaning that its value always increase over time. This is achieved by, among other things, including the timestamp of the primary database's transaction log page (derived from the ASE database's internal timestamp counter) in the OQID.

Replication Server relies on the fact that the OQID is universally increasing: if its sees a transaction whose OQID is lower than a previously processed transaction, this means the primary database is re-sending earlier transactions, for example because a database dump was restored. Since those earlier transactions have already been replicated, they should not be delivered to the replicate databases again, and Replication Server will therefore silently ignore those transactions as duplicates. This aspect of Replication Server is known as "duplicate detection".

To implement duplicate detection, Replication Server records the highest OQID processed from each primary database in its stable queues. Whenever a transaction shows up with a lower OQID, it is considered a duplicate and discarded.

Duplicate Detection and IMDB replication

When replicating from a primary IMDB/RDDB, duplicate detection in its normal form would stop replication from working correctly. This is because the primary IMDB/RDDB is initialized from the template database after every ASE restart. As a result, the IMDB/RDDB finds itself in the same state after every ASE reboot, identical to the first time it was created, with the same secondary truncation point and the same database timestamp, since both are copied from the template database (where we assume no update activity). This has the effect of generating replicated transactions with the same OQID values as after the previous ASE restart. Without any adaptations, Replication Server will therefore discard all these primary transactions until the point is reached where the OQID exceeds an earlier high-water mark.

Since this is an unacceptable side-effect, duplicate detection needs to be disabled for primary IMDB/RDDB databases. This happens automatically in ASE 15.5 ESD#4 and later: when the ASE Replication Agent starts in a IMDB/RDDB `with durability=no_recovery`, it sends a **'resetqueue'** command to Replication Server which resets all downstream duplicate detection for transaction originating from that primary database. This is done by resetting the highest OQID seen previously from this primary database, to 0.

This **'resetqueue'** command is only sent when the ASE Replication Agent starts for the first time after the IMDB/RDDB was created; a subsequent stop and start of the IMDB Replication Agent will not cause another **'resetqueue'** command to be sent.

Notes:

- Note that the **'resetqueue'** command is not logged in the Replication Server errorlog file, making it transparent for DBAs (though interested DBAs can observe it by dumping the contents of the stable queues).
- The **'resetqueue'** command has no effect on committed transactions originating from the primary IMDB that are still in the Replication Server queues and have not yet been applied to the replicate database. These transactions will be applied to the replicate database normally.
- For completeness, the **'resetqueue'** command also removes any uncommitted open transactions from this primary database from the stable queues, to avoid errors in certain scenarios. This has no impact on functionality, since such uncommitted transactions would not have been applied to the replicate database anyway.
- A **'resetqueue'** command is sent by the ASE Replication Agent for an IMDB or RDDB with **durability=no_recovery**. For an RDDB with **durability=at_shutdown** or for a regular database with **durability=full,** no **'resetqueue'** command is sent since these databases are 'normal' database from a replication viewpoint since they provide persistent storage across ASE restarts.
- The **'resetqueue'** command was in fact implemented before ASE 15.5 ESD#4, but for technical reasons, this is supported only in version 15.5 ESD#4 (or later).

Primary IMDB with Replication Server pre-15.5 (unsupported)

Replication from a primary IMDB is only supported with Replication Server 15.6 ESD#1 or later. For completeness, the following unsupported combination with earlier versions of Replication Server is included in this paper.

The **'resetqueue'** command was implemented in Replication Server 15.5. When using a primary IMDB/RDDB with an earlier version of Replication Server than 15.5, the ASE Replication Agent will detect this and will not send a **'resetqueue'** command. Instead, it will increment the primary database's generation ID instead. This also has the effect of not discarding new transactions as duplicates, since the generation ID is also part of the OQID.

However, since the primary DB generation ID is a 2-byte value, it will run out after 65535 increments. Should that happen, Replication Server will generate an error, and the primary database connection must be dropped and recreated (as well as all replication definitions and subscriptions associated with this primary database). Also, the generation number in the primary database will need to be set back to 0 manually.

To avoid issues with the generation ID running out, as well as a number of additional internal issues, replication from a primary IMDB/RDDB is only supported with Replication Server 15.6 ESD#1 (or later).

Appendix: Code Examples

Assumptions in these examples:

- name of primary dataserver: **PRIMDS**
- name of replicate dataserver: **REPDS**
- name of primary IMDB: **prim_imdb**, **prim_imdb2**
- name of template database: **imdb_template**

Setting up a single primary IMDB with a template DB

This is an example of setting up replication from a single IMDB using a template DB.

In ASE:

```
-- This will become the template DB, but first set it up with the
-- name of the future IMDB
use master
go
create database prim_imdb
on my_disk_device = '50M'   -- template does not need to be large
with durability=full    -- this is the default, may be omitted
go

use prim_imdb
go
-- Create the primary table 'primtab' and mark it for replication:
create table primtab (pk int unique, attrib varchar(50))
go
sp_setreptable primtab, true
go

-- Or mark the entire database for replication:
sp_reptostandby prim_imdb, 'all'
go
```

In Replication Server:

```
-- Run rs_init to create a connection to prim_imdb, as
-- a replicated primary database
-- Either an interactive rs_init session (not shown here)
-- or use a resource file:
--     rs_init -r <resource-file-name>
```

For table replication:

```
-- Create a table repdef for 'primtab'
create replication definition primtab
with primary at PRIMDS.prim_imdb
with replicate table named reptab
(pk int, attrib varchar(50))
primary key(pk)
go

-- Create subscription (assuming the replicate DB connection
-- already exists)
create subscription reptab
for primtab
with replicate at REPDS.repdb
without materialization  -- choose your materialization method
go
```

For database replication:

```
create database replication definition prim_imdb
with primary at PRIMDS.prim_imdb
replicate ddl  -- optional; when using DDL replication, you must
               -- enable 'send warm standby xacts' in the RepAgent!
go

create subscription sub_prim_imdb
for database replication definition prim_imdb
with primary at PRIMDS.prim_imdb
with replicate at REPDS.repdb
without materialization
go
```

In ASE:

```
-- Verify replication works by inserting a row in the primary table
-- and observe the transaction being replicated correctly
insert prim_imdb..primtab values (1, 'test row 1')
go


-- Assuming replication works fine,stop the RepAgent
use prim_imdb
go
sp_stop_rep_agent prim_imdb
go
```

In Replication Server:

```
-- Suspend all connections to prim_imdb
```

```
suspend connection to PRIMDS.prim_imdb
go
suspend log transfer from PRIMDS.prim_imdb
go
```

In ASE:

```
-- Rename the current database to the template's name
use master
go
sp_dboption prim_imdb, single, true
go
sp_renamedb prim_imdb, imdb_template
go
sp_dboption imdb_template, single, false
go

-- Optional: mark the RepAgent as started in the template if it must
-- start automatically in the primary IMDB after an ASE restart
use imdb_template
go
-- ASE 15.5 ESD#5 or later:
sp_config_rep_agent imdb_template, 'auto start', true
go
-- ASE 15.5 ESD#4:
sp_start_rep_agent imdb_template
go


-- Now create the IMDB that will be the real primary database
sp_cacheconfig 'imdb_cache', '1G', inmemory_storage
go
disk init name='imdb_device', physname='imdb_cache',
size='1G', type=inmemory
go

create inmemory database prim_imdb
use imdb_template as template
on imdb_device = '1G'
with durability=no_recovery
go

-- Start the RepAgent:
use prim_imdb
go
sp_start_rep_agent prim_imdb
go
```

In Replication Server:

```
resume connection to PRIMDS.prim_imdb
go
resume log transfer from PRIMDS.prim_imdb
go
```

In ASE (optional):

```
-- Alias the maintenance user to 'dbo' in the template database
-- This is relevant only when the IMDB will also be used as a
-- replicate database
use imdb_template
go
sp_dropuser prim_imdb_maint
go
sp_addalias prim_imdb_maint, dbo
go
```

At this point, replication from the primary IMDB works normally.
After stopping & restarting ASE, the ASE Replication Agent in the primary IMDB is either started automatically (if it was configured as started in the template DB), or it must be started manually.

Setting up an additional primary IMDB/RDDB with an existing template DB

This is an example of setting up an additional IMDB (name **prim_imdb2**) as a primary database based on an existing template database that is already used for another primary IMDB.

In ASE:

```
-- Create the database, but *without* a template
-- Also, it can be a regular full-durability DB at this point
create database prim_imdb2 on dev1='10M'
go
```

In Replication Server:

```
-- Run rs_init to create a connection to prim_imdb2, as
-- a replicated primary database
-- Either an interactive rs_init session (not shown here)
-- or use a resource file:
--      rs_init -r <resource-file-name>
```

In ASE:

```
-- Stop the RepAgent and drop database prim_imdb2
use master
go
sp_stop_rep_agent prim_imdb2
go
```

```
drop database prim_imdb2
go


-- Now recreate the DB, but as a real IMDB, and with the template DB
sp_cacheconfig 'imdb2_cache', '1G', inmemory_storage
go
disk init name='imdb2_device', physname='imdb2_cache',
size='1G', type=inmemory
go

create inmemory database prim_imdb2
use imdb_template as template  -- this template DB already exists
on imdb2_device = '1G'
with durability=no_recovery
go

-- Start the RepAgent:
use prim_imdb2
go
sp_start_rep_agent prim_imdb2
go
```

In Replication Server:

For table replication:

```
create replication definition primtab2
with primary at PRIMDS.prim_imdb2
with primary table named primtab  -- 'primtab' exists in template DB
with replicate table named reptab
(pk int, attrib varchar(50))
primary key(pk)
go

create subscription reptab2
for primtab2
with replicate at REPDS.repdb
without materialization  -- choose your materialization method
go
```

For database replication:

```
create database replication definition prim_imdb2
with primary at PRIMDS.prim_imdb2
replicate ddl  -- optional; when using DDL replication, you must
               -- enable 'send warm standby xacts' in the RepAgent!
go

create subscription sub_prim_imdb2
for database replication definition prim_imdb2
```

```
with primary at PRIMDS.prim_imdb2
with replicate at REPDS.repdb
without materialization
go
```

In ASE:

```
-- Add the maintenance user in the template database
-- Optionally, this user can be added as a dbo alias
use imdb_template
go
-- Choose only one of the following lines:
sp_dropuser prim_imdb2_maint
sp_addalias prim_imdb2_maint, dbo
go


-- Now start the RepAgent
use prim_imdb2
go
sp_stop_rep_agent prim_imdb2
go

-- Replication from prim_imdb2 works now
```

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