How to improve the performance of the SUP Cache Database (CDB)

Applicable Release:
SUP 2.2

Version 1.0
March 2013
SAP “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Authored By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.90</td>
<td>Marvin Hoffmann</td>
<td>Wiki article</td>
</tr>
<tr>
<td>0.90</td>
<td>Patrick Kelleher</td>
<td>Document preparation</td>
</tr>
<tr>
<td>1.0</td>
<td>Patrick Kelleher</td>
<td>Corrections, additional content</td>
</tr>
</tbody>
</table>
**Typographic Conventions**

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Text</td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td>Example text</td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td>Example text</td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td>Example text</td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td>&lt;Example text&gt;</td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td>EXAMPLE TEXT</td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

**Icons**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚠️</td>
<td>Caution</td>
</tr>
<tr>
<td>📜</td>
<td>Note or Important</td>
</tr>
<tr>
<td>📚</td>
<td>Example</td>
</tr>
<tr>
<td>🏷</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
## Table of Contents

1. Introduction .......................................................................................................... 1
2. Tuning Database Performance .............................................................................. 2
3. Routine Database Optimization ............................................................................ 5
4. Appendix ................................................................................................................ 11
1 Introduction

This document covers performance optimization of SQL Anywhere databases. The Sybase Unwired Platform (SUP) contains a server side cache called the Cache Database (CDB), which is a SQL Anywhere database instance. The CDB is unique in its maintenance requirements; we will now look at why that is.

In SUP, mobile data architects take requirements from the business, UI designs from the mobile development team, and data interface definitions from the SAP basis team. These architects then use this information to build Mobile Business Object (MBO) data models using the Sybase Mobile SDK. When these MBO’s are packaged and deployed to SUP, they are used to generate tables in the CDB. During normal operation, the cache database is continuously executing batch transaction based on changes it received from multiple mobile devices and from one or many EIS systems.

The result of this constant turbulence on the CDB is that the data and tables become fragmented and disorganized, indexes become sparse and redundant data can remain lost among the rest of the data. All of these side effects can degrade performance, if you do not take active steps to resolve them.

Steps a database administrator must take include:

- Tuning the database for improved performance by removing redundant data from the CDB and other SUP databases.
- Routine Database optimization such as addressing table fragmentation and index issues accordingly.
## 2 Tuning Database Performance

There are some simple settings you can configure that will help you ensure that the CDB is running optimally.

### Preclude virus scanning of CDB and other database files

Removing the database files from your company's virus scanner's "on access" scans will improve performance and stability of the SUP database. Ensure the following folders are excluded from the scan (<d:> refers to the virtual drive the SUP database files are installed to):

- `<C:\Sybase\UnwiredPlatform\Servers\UnwiredServer\data`
- `<C:\Sybase\UnwiredPlatform\Servers\SQLAnywhere12\data`

### Remove redundant data from the CDB

In the SAP Mobile Platform the cache database should be cleaned up regularly to support system performance.

You can automate domain-level cleanup based on a configured schedule for specific cleanup categories. Running the cleanup options uses system resources, so SAP recommends that you schedule these tasks when system load is lowest.

The following table consists of a summary of the types of domain-level cleanup options available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription Cleanup</td>
<td>Removes subscriptions that are no longer referenced by any active users.</td>
</tr>
<tr>
<td>Online Cache Cleanup</td>
<td>Removes mobile business objects (MBOs) from the cache group that uses the Online cache policy.</td>
</tr>
<tr>
<td>Error History Cleanup</td>
<td>Removes historical data on MBO data refresh and operation replay failures.</td>
</tr>
<tr>
<td>Client Log Cleanup</td>
<td>Removes client log records that have already been synchronized to the device, or are no longer associated with active users.</td>
</tr>
<tr>
<td>Synchronization Cache Cleanup</td>
<td>Removes logically deleted rows in the cache that are older than the oldest synchronization time on record in the system.</td>
</tr>
</tbody>
</table>

For detailed information refer to the online help (http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc01092.0224/doc/html/mqu1285284377262.html).
Tuning the initial cache sizes

The CacheDB is the default database of the SUP and is heavily used:

- In a non-cluster installation the CDB is stored under `C:\Sybase\UnwiredPlatform\Servers\SQLAnywhere12\data` as default.db.
- In a cluster installation the database file are stored on the data tier host under `C:\Sybase\UnwiredPlatform\Data\CDB`. The default minimum cache size of this database is 24MB. If this 24 MB is too small the CDB will dynamically adapt the cache size.

There are two properties associated with an initial cache size that are dynamically adjusted by the system as needed: minimum and maximum cache size. This automatic adjustment of cache size will increase the CPU processing time and the Disk I/Os.

- The maximum initial cache size you see in the logs is calculated dynamically based on the physical memory available on the server. This is the maximum size that the cache will grow to under load. If there are memory constraints on the server this can be limited by setting the `-ch` option in the `SUP_HOME\Servers\UnwiredServer\config\configure-sup.xml`. Sybase recommends that you not set `-ch`; instead make sure there is enough RAM available to accommodate the peak cache size.
- The minimum initial cache size, by default, will be set to the initial cache size (the `-c` option described below). This can be modified using the `-cl` option in the `configure-sup.xml` if required.

Hint: If the cache size is adapted using this setting, this modification is always logged into the `errorlog.txt` located in the log directory.

If you have a deployed an MBO package that is using a huge amount of data, increase the initial cache size to avoid many dynamic adjustment operations:

1. As a general recommendation, set the initial minimum CDB size to at least 50% of available RAM (Sybase recommendation: 8 GB)
2. On startup, by default, the minimum cache size should be 28408K, and the maximum cache size should be 7545652K. It will adjust itself dynamically.
3. Go to `C:\Sybase\UnwiredPlatform\Servers\SQLAnywhere12\data\` or in a Cluster environment to `<drive>\Sybase\UnwiredPlatform\Data\CDB`
4. Edit `install-sup-sqlany12.bat`
5. Perform a text search for SharedDB
6. Replace the parameter `-c 24M with -c 70P` IMPORTANT

The documentation states that you can use `-c 70%%` instead of `-c 70P`. We found that this is dangerous. In the batch file (`install-sup-sqlany12.bat`) you can use this, but in the ini file (`cdboptions.ini`) the escaped character `%` is not recognized, so here only `-c 70%` is allowed. Because the ini file gets overwritten in the batch file, we recommend always using `-c 70P`.

7. Go to `C:\Sybase\UnwiredPlatform\Servers\UnwiredServer\config`
8. Edit `configure-sup.xml` and change value also to `-c 70P`
9. Check cache size value and other properties. You can do this, by checking the `errorlog.txt`. Always when the database is starting this information is logged. You can also do this by calling the stored procedure: `sa_eng_properties`
3 Routine Database Optimization

In order to maintain database performance it is essential to run regular maintenance scripts against the database. These operations optimize the performance of the database: often, whole sets of records are added, duplicated, updated or dropped as the cache is synched and refreshed.

1. When records are deleted from a table, the table index for that record now refers to an empty record. If this occurs many times over the lifetime of the application, then data retrieval will be affected as the index becomes sparse.

2. When sets of data are deleted, such as when device data is dropped from the cache, whole sets of indexes are removed. This is analogous to having a set of encyclopedias and then removing whole topics form the set. As a result you still have the same number of volumes, and each volume will have the same range of page number, but some pages will be missing. When you search for a topic it still takes almost as long as when all the pages were still there. If new topics are added then they will be added to new volumes so the set will grow even if the number of topics does not.

Reorganizing the table will reduce the larger gaps in the table but will not re-index the entire table.
Rebuilding the database creates a new index thus removing all gaps including those left by single record deletion.

The following maintenance steps are covered below:
- Rebuilding Platform Databases
- Defragmenting the database
- Rebuilding the CDB
- Database rebuilds and transaction logs
- Database rebuilds and downtime

Note: these operations require the database to be offline and thus force downtime on production servers.

**Rebuilding Platform Databases**

Rebuilding the database will export the database data to a temp directory, and then reloading the data into the same database, thus recreating the entire database index. Note, usually the data is exported to the C:\Temp directory by default so ensure you have sufficient access rights on the root directory if you are not going to specify another location (see options).

Rebuild platform databases regularly. Otherwise, the database log can grow too large (several gigabytes in size) to act as a proper cache, and performance could degrade. If you experience long shutdown times with data services for the Unwired Platform data tier, you need to unload and reload data using the dbunload utility. This rebuilds your database and maintains a healthy data tier performance. Sybase recommends that you perform this action every four or five months. Stop all runtime services (data tier and server nodes), including the cache database, the cluster database, and synchronization services. Perform a full off-line backup of each, by copying the database and transaction log files to a secure location. See Backing Up SQL Anywhere Databases. Rebuild each runtime database with the SQL Anywhere Unload (dbunload) utility. The dbunload utility is available in the directory:

<UnwiredPlatform_InstallDir>\Servers\SQLAnywhere11\BIN32

Validate the data before restarting the services.
Defragmenting the database

Before evaluating the need to rebuild the database ensure that you have checked the disk fragmentation and are content that disk fragmentation is not an issue.

Beside the disk fragmentation a running database will be also fragmented inside the database file. If a database is heavily fragmented, SQL queries need more time for their execution. The CDB is especially susceptible to fragmentation due to the high volume of data turnover, for example when deploying and deleting packages.

To check the current fragmentation of CDB, connect via dbisql (C:\Sybase\UnwiredPlatform\Servers\MessagingServer\Bin\dbisql.exe) to the corresponding database and call the following stored procedures:

\textbf{sa\_table\_fragmentation}

\begin{tabular}{|l|c|c|c|}
\hline
\textbf{tablename} & \textbf{rows} & \textbf{row\_segments} & \textbf{segs\_per\_row} \\
\hline
\texttt{SAMPLE\_1\_0\_ADLGROUP\_CACHE\_LOCK} & 0 & 0 & 0 \\
\texttt{SAMPLE\_1\_0\_CACHE\_POLICY} & 4 & 4 & 1 \\
\texttt{SAMPLE\_1\_0\_CACHE\_STATE} & 4 & 4 & 1 \\
\texttt{SAMPLE\_1\_0\_CACHE\_UPDATE} & 4 & 4 & 1 \\
\texttt{SAMPLE\_1\_0\_MASTERGROUP\_CACHE\_LOCK} & 0 & 0 & 0 \\
\texttt{SAMPLE\_1\_0\_PARTITION} & 842 & 842 & 1 \\
\texttt{SAMPLE\_1\_0\_PARTITION\_REFRESH} & 807 & 807 & 1 \\
\texttt{SAMPLE\_1\_0\_SUPERMASTERGROUP\_CACHE\_LOCK} & 0 & 0 & 0 \\
\texttt{SAMPLE\_1\_0\_TRANSACTIONALGROUP\_CACHE\_LOCK} & 0 & 0 & 0 \\
\texttt{SAMPLE\_1\_0\_acc\_27522} & 202 & 202 & 1 \\
\texttt{SAMPLE\_1\_0\_acc\_66455} & 53,544 & 58,793 & 1.098031525474376 \\
\texttt{SAMPLE\_1\_0\_acc\_66455\_sk} & 23,969 & 23,969 & 1 \\
\texttt{SAMPLE\_1\_0\_acc\_69235} & 202 & 202 & 1 \\
\texttt{SAMPLE\_1\_0\_acc\_96862} & 53,544 & 54,911 & 1.025530404900642 \\
\hline
\end{tabular}

If the number of rows and number of row segments is the same (segs\_per\_row is 0 or 1) then there is no fragmentation. Higher segs\_per\_row number is worse.

The table indexes can also be affected by the constant data swapping that occurs in the CDB. As a result an index can end up containing many gaps in its range. The index density is a measure of how sparse or dense the range of an index is, i.e. how few gaps the range contains. The \texttt{sa\_index\_density} stored procedure returns the density for each index in the CDB.
How to Improve the Performance of the SUP Cache Database

IndexType: PKEY = Primary Key, NUI = Non Unique Index, UI = Unique Index
Density = 1 is perfect, if there are a lot of LeafPage density values down to 0.7 are acceptable.
Values below 0.7 can affect the performance
Skew = 1 is perfect, higher is worse

If the index density is below an acceptable density then the SQL command REORGANIZE TABLE can be used to rebuild the table’s index.

Example:

REORGANIZE TABLE <db_name> primary key
exec sa_index_density <db_name>

Results: (Execution: 16,678 Seconds)
Before Reorg: LeafPages 15,656 and Density: 0.5734  After Reorg: LeafPage 7,825 and Density: 0.9915

For a simple method of evaluating if you need to look at database defragmenting (say as a prerequisite to the above procedure) look at the article “When to reclaim CDB space in SUP 2.x”: https://scn.sap.com/docs/DOC-39547

Rebuilding the CDB (dbunload)

If your database suffers from extensive fragmentation, you can fix this with the tool dbunload.
Sybase recommends that you perform this action every four or five months.

1. Stop all SUP nodes
2. Stop SQLAnywhere services (CacheDB, ClusterDB, MonitorDB)
3. Open cmd on C:\Sybase\UnwiredPlatform\Servers\SQLAnywhere12\BIN32
4. Execute:
   dbunload -c
   "DBF=C:\Sybase\UnwiredPlatform\Data\CDB\default.db;UID=dba;PWD=sql" -ar
5. Optionally, you can choose to rebuild the database into a new database by using ther –an option:
   dbunload -c
   "DBF=C:\Sybase\UnwiredPlatform\Data\CDB\default.db;UID=dba;PWD=sql" –an C:\temp\default.db
After the command is executed:

- The file size decreases (for example a DB of 7GB might shrink down to 1GB after the execution of this command).
- The database log files will be deleted.
- The noticeable performance improvement should be observed, and should in proportion to the relative size decrease of the DB file.

Note, the –ar option in the second example above will:

- Create a new database with the same settings as the old database,
- Reload the database, and
- Replace the old database.

However, you can change the initialization options as necessary by specifying other supported dbunload options (such as -ap to change the page size or -et to enable table encryption).

For more information on the dbunload utility, please refer to the online documentation:

Database Rebuilds and Transaction Logs

A transaction log is created for recording all CDB transactions in case the database fails and has to be restored. In order to prevent the transaction log from growing indefinitely, use the dbbackup utility to incrementally update your offline backup of the CDB or use the BACKUP DATABASE SQL command to create a stored procedure that regularly performs this incremental backup.


When rebuilding the database, the new database can no longer be restored using the transaction logs written prior to the build. Therefore it is important to perform the following before rebuilding the database:

1. Stop all SQL Anywhere services.
2. Make full backups of the platform databases and transaction logs.
3. Delete the transaction log of the CDB (default.log).

Also, make sure to create a backup of the rebuilt database once the rebuild has completed successfully. This will be used in the future if the database will need to be restored form the transaction logs.

Once the CDB is restarted it will write a new transaction log based on the rebuilt database will be created by the system. If you are using transaction log mirroring, ensure that this is still running, as it might have to be reinitialized.
Database Rebuilds and Downtime

The total downtime for the platform can be expensive as it consists of:

- Stopping all nodes in the cluster
- Stopping all data tier services
- Performing a full database backup
- Performing a database rebuild
- Performing another database backup
- Restarting the Data Tier services
- Restarting all nodes in the cluster

You can minimize this downtime by performing the following steps:

1. Retrieve a backup of the CDB from storage or use the dbbackup utility to create a new backup
2. Run dbbackup using the –r option to truncate the existing transaction log
3. Restore the backup CDB to the latest transaction log created in step b
4. Move the restored backup CDB to a separate computer and rebuild the database
5. Shut down the cluster and data tier
6. Run dbtran on the transaction log to create an SQL output file that can be applied to the rebuild CDB
7. Backup the current CDB and replace the default.db and default.log with the rebuilt default.db file.
8. Start the data tier but not the cluster nodes.
9. Apply the SQL output from step f to the CDB
10. Start the cluster nodes

You can further reduce the overall downtime by reducing the need to rebuild the database. If you monitor the table index density regularly and determine which tables are fragmented, you can run regular maintenance on just these tables during off-peak hours.

Reorganizing the table will lock the table and cause users trying to get write access to the table to get errors but nevertheless, this can be performed online. Reorganizing the tables individually is not as effective as rebuilding the entire database but it will reduce the frequency a complete rebuild is required.
## Appendix

<table>
<thead>
<tr>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
</table>