

Improving the Performance of ERwin Model Manager

Introduction

The performance of Model Manager (V4, V7 or V8) depends on a number of parameters. While CA have stated that a major focus of R9 of the ERwin suite will be a major upgrade of Model Manager there are options which can be used in the current versions (V4, V7 and R8) to maximize the performance of the application.

The most common reason for performance degrading is the volume of data in the repositories database, a database which was not designed to host the major increase in data volumes which has been witnessed in recent years.

The major increase in data volumes arises from a number of factors including:

- The number of ERwin models being managed by users
- The size of ERwin models (Models with 1000+ entities/tables are now commonplace!)
- The increased amount of meta data arising from increased DBMS support in ERwin

The increased levels of data in the repository focus on two specific tables.

- m7objectproperty (mobjectproperty in V4 and m8objectproperty in R8)
- m7object (mobject in V4 and m8object in R8)

Tests have shown that when the number of rows in table m7objectproperty exceed 20 million performance will start to degrade.

The topics which need to be addressed when trying to improve repository performance include:

- The server and client machines
- The repository setup
- The repositories maintenance regime
- Whether single or multiple repositories are used

The Server

The specified minimum server requirements are just that and bear no resemblance to the server needed to support a large repository with many users. Possibly of greater significance with regard to the server specification is the requirements defined by the DBMS vendor. That and the I/O intensive nature of the Model Manager repository (see section on disc storage) are the real server issues.

Whilst the ideal situation is to use a dedicated server this will often raise other management and possibly DBMS licensing issues. Where performance is an issue it is recommended that the server CPU,

memory usage and disc I/O be monitored to identify applications which are competing for the same resource.

The Model Manager Administrator Guide which is installed on the client contains dbms tunings that may be applicable.

Client workstation

The client PC has a significant impact on Model Manager performance.

- RAM. The most important factor for the client PC is the RAM memory. It is recommended that client workstations have at least 2GB of RAM, however, that is a minimum requirement only.
- Clearly the higher the speed of the client the better the performance in processing the large amounts of data which are read from or written to the repository.
- Disk Space and Virtual Memory. Make sure you have enough disk space available and set the Virtual Memory to 1.5 times the size of your RAM memory or more. Make sure the Initial Size and the Maximum Size of the Paging File is set to the SAME size. Why? If the initial and maximum sizes differ, the system must expand and shrink the paging file as needed and that affects performance.

The Network

While most corporate networks are fast with considerable bandwidth opening a large model can involve 50MB of data. Any performance hit arising from local networks is usually small in comparison with other issues.

Disc Storage

When the repository is hosted on Oracle the recommendation is for the database files to be split across 4 discs.

It should also be noted that the Model Manager application uses a significant amount of I/O resource and its data files should not be hosted on discs where it will compete for resource with other applications which are also I/O intensive.

While it is not part of the standard installation it is possible to create table specific tablespaces (or Filegroups for SQL Server) and to place their data files on different discs thereby increasing the I/O resource.

Daily Maintenances tasks

Re-indexing

Due to the manner in which Model Manager works its indexes very quickly become unbalanced.

CA recommends that for large repositories where performance is an issue the repository indexes are rebuilt daily.

DB Statistics

Model Manager's supported DBMS all utilize database statistics to formulate the execution plans for queries and CA recommends that for large repositories where performance is an issue the database statistics be updated daily.

Periodic maintenance tasks

Purging models

The number of model versions held in a repository quickly builds up if unwanted versions are not purged.

Model Manager best practice recommends that baseline versions of models which have been approved at critical project milestones be 'Marked' as such and retained while all other versions are purged.

At the end of each month any unmarked versions for the previous month are then purged leaving all working versions for at least one month available.

Some users also 'Mark' each models last version of each month for retention.

Upgrading the version of Model Manager

There have been some version specific performance issues which can be addressed by upgrading to a later version. V7.3.3 is a case in point. Any V7 repository earlier than V7.3.3 should be upgraded.

Oracle only

Some clients have added Oracle hints reflecting their own environment to the queries contain in Model Manager Procedures. This subject is beyond the scope of this document.

Oracle's Cost-Based Optimizer

Consider turning on Oracle's Cost-Based optimizer using system statistics.

For further information see the article 'Understanding System Statistics'

http://www.oracle.com/technology/pub/articles/lewis_cbo.html

Additional Indexes

Adding additional indexes may speed up Model Manager performance in some, but not all cases. The following index is recommended by CA where there are performance issues.

Oracle:

```
CREATE INDEX XAK3_MASTVERS ON m7object
(
  MASTERID ASC,
  STARTVERSION ASC
)
LOGGING
TABLESPACE <DATA_TABLESPACE>; <--- Replace with the name of your Data
Tablespace (e.g. MMADMIN).
```

SQL Server:

```
CREATE INDEX XAK3_MASTVERS ON m7object
(
  MASTERID          ASC,
  STARTVERSION      ASC
)
go
```

Subject areas and stored displays

ERwin diagrams are held as raw positional data and the application generates graphics from that raw data when the model is opened. In large models the generation of these graphics can cause significant delays when opening the model. There are three factors which need to be considered.

- The number of subject areas
- The number of entities/tables and relationships in the subject area
- The number of stored displays for the subject area

Maintaining multiple stored displays for the main subject area of large models is the worst case scenario.

Remove History Objects

By default ERwin records some historical information relating to the Model, Entities/Tables and Attributes/Columns. This historical information is not critical to the model and is rarely used. To see what information can be recorded check the dialogue at Model > Model Properties>History Options where the option can also be turned off.

Turning off the option will stop new history records being created but will not remove the existing history information from the model. It is possible however to delete the history objects from the repository either by model, by library or for the whole repository.

In extreme cases the history records can form a significant proportion of the models data and deleting it from the repository can sometimes have a positive impact on the repositories performance.

Run this query to get a count of the history objects in the mart:

```
select c.ClassName,count(o.classid) as ClassCount from m7object o,m7class c
  where o.classid = c.classid and o.classid=1075839089 group by
  o.classid,c.classname
 order by ClassCount desc;
```

To delete all history objects from the mart use the following script (Backup first!):

```
DELETE FROM m7ObjectProperty WHERE ObjectID IN
  (SELECT ObjectID FROM m7Object WHERE ClassId = 1075839089 );
DELETE FROM m7Object WHERE ClassId = 1075839089;
Commit;
```

Use multiple marts.

Note that when using multiple marts a licence can be used by a single individual to access multiple marts but cannot be used by different individuals on different marts.

Multiple marts can be used in a number of ways but they all effectively partition the models in some manner across multiple repositories. The scenarios for running multiple marts include:

Partition the models by user, team, application or organisational unit etc.

With this approach the models relating to different teams, or applications or organisational units would be held in a repository from that category.

Retain the existing repository as an archive repository and create a new repository for active work.

This approach may be appropriate where a repository is in need drastic maintenance to reduce the volume of data. The alternative is to create a new repository which will be used and maintained from that point forward. Only the latest version of each model will be moved to it and the historic data will still be available in the archive repository

Auto purging unwanted versions

Unwanted versions can be purged from the Model Manager Version Manager but a stored procedure is also provided in the repository to purge unmarked versions from the repository. The version for V7 is shown below.

```
m7x_PurgeVersion_Mart(n)
```

Replace the parameter 'n' with the number of versions of each model to be retained and the last 'n' unmarked versions of each model will be retained. All other unmarked versions will be deleted. Marked versions will not be removed. Any versions to be specifically retained should be 'Marked' before executing this procedure. (The database should also be backed up!)

Re indexing Scripts

The Oracle DBMS reindex script(Oracle V7.3 or later):

How to use the reindexing script:

1. Copy the Script mmreindex.ora locally (here it is assumed to be the C: drive).
2. Edit the script and replace 'MODELMART' with the name of the AllFusion Model Manager schema-owner and also replace 'MMINDEX' with the name of the AllFusion Model Manager index tablespace.
3. Connect to your Oracle query tool as the Schema Owner (who will need ALTER ANY INDEX privileges) .

Execute the following Script at the SQL Prompt,

```
@c:\mmreindex.ora
```

Troubleshooting the reindexing script.

If your query tool responds:

no rows selected

not spooling currently

The term 'MODELMART' in the script has not been correctly replaced with the name of the Model Manager schema-owner.

The Oracle dbms reindex script (MMReIndex.ora):

```
-----
-- Object: MMReIndex.ora
-- Desc: Use this Procedure to ReIndex the MM ORACLE Repository whenever a
-- Merge/Save of big model is done to MM
-- Limitation(s) is specific to Oracle Releases >= 817
-- For ORACLE DBMS < 8i Modify the script to Use NOPARALLEL
-- NOTE: You will need to change MODELMART to the Model Manager schema owner name.
-- You will need to change MMINDEX to the Model Manager index tablespace.
-- Oracle indexes are not self-balancing. They become fragmented after a large
-- number of INSERTs and DELETE which may lead to significant performance degradation.
-- This script rebuilds the Model Manager indexes and cures them.
-----
```

```
-----
set pagesize 1000
set linesize 2000
set verify off
set feedback off
set heading off
spool c:\mmreindex.ora
SELECT 'ALTER INDEX ' || USER || '.' || INDEX_NAME ||
' REBUILD PARALLEL NOLOGGING COMPUTE STATISTICS TABLESPACE MMINDEX;'
FROM DBA_INDEXES
-----
```

```

WHERE OWNER = UPPER ('MODELMART')
AND (INDEX_NAME like 'XPK%' or INDEX_NAME like 'XAK%'
     or INDEX_NAME like 'XIE%')
order by index_name;
spool off
set heading on
set pagesize 24
set verify on
set feedback on
@c:\mmreindex.ora
/

```

The MS SQL Server dbms reindex script:

```

-- Drop the Procedure appropriately
IF EXISTS (SELECT name FROM sysobjects WHERE name = N'usp_ReIndex' AND type = N'P')
Begin
    DROP PROCEDURE usp_ReIndex
    Print 'Procedure Dropped'
End
GO

-----
-- Object:  usp_ReIndex
-- Desc:   Use this Procedure to ReIndex the MM SQL REpository whenever a
--         Merge/Save of big model is done to MM
-- Change History:
-- Name      Date      Reason
-----
CREATE PROCEDURE usp_ReIndex AS
    Declare
        @Cmd varchar(2000),
        @Name Sysname
    DECLARE tmp_Reindex CURSOR LOCAL FOR
        SELECT Name
        FROM SysObjects
        WHERE Type = 'U'
    OPEN tmp_Reindex
    FETCH NEXT FROM tmp_Reindex INTO @Name
    WHILE @@FETCH_STATUS = 0
    BEGIN
        Print 'Processing Index for Table ' + @Name
        Set @cmd = 'DBCC DBREINDEX (' + @Name + ', "", 0)'

```

```

Exec (@Cmd)
If @@Error <> 0
    Print 'Error Reindexing Table ' + @Name
    FETCH NEXT FROM tmp_Reindex INTO @Name
END
Close tmp_Reindex
go
-- ReIndex the DB
Exec usp_ReIndex
The Sybase dbms reindex script:
-- Drop the Procedure appropriately
IF EXISTS (SELECT name FROM sysobjects WHERE name = N'usp_ReIndex' AND type = N'P')
Begin
DROP PROCEDURE usp_ReIndex
Print 'Procedure Dropped'
End
GO
-----
-- Object: usp_ReIndex
-- Desc: Use this Procedure to ReIndex the MM SQL REpository whenever a
-- Merge/Save of big model is done to MM
-- Change History:
-- Name Date Reason
-----
CREATE PROCEDURE usp_ReIndex AS
Declare
@Cmd varchar(2000),
@Name Sysname(100) ,
@output_str varchar( 255 )
DECLARE tmp_Reindex CURSOR FOR
SELECT name
FROM sysobjects
WHERE type = 'U'
OPEN tmp_Reindex
FETCH tmp_Reindex INTO @Name
WHILE ( @@sqlstatus = 0 )
BEGIN
SELECT @output_str = 'Processing Index for Table '+@Name
Print @output_str
--Set @Cmd = 'DBCC REINDEX (' + @Name + ')'
--Exec (@Cmd)
DBCC REINDEX (@Name )
If @@Error <> 0

```

```
Begin
SELECT @output_str = 'Processing Index for Table1 '+@Name
Print @output_str
end
FETCH tmp_Reindex INTO @Name
END
Close tmp_Reindex
go
EXEC usp_ReIndex
```