

Microsoft SQL Server to ASE Migration Guide for Microsoft SQL Server V7.0 and ASE 11.9.2

*Sybase, Inc.
30 September 1998.*

Abstract

This document is intended to be an aid to those customers who need to migrate their applications from Microsoft SQL Server to Sybase Adaptive Server Enterprise or who need to ensure application portability between the products. It explains the differences between the products and presents a migration process to assist a customer with migration from Microsoft SQL Server. It also provides checklists to assist with migration and application portability between the two products.

Copyright © 1998 Sybase, Inc. All rights reserved.

This White Paper is for informational purposes only. Sybase makes no warranties, express or implied in this document. The information in this document is subject to change without notice.

Sybase, Adaptive Server, Backup Server, Replication Server, OmniConnect and Sybase Central are trademarks of Sybase, Inc. 6/97.

All other company and product names used herein may be the trademarks or registered trademarks of their respective companies.

Sybase Incorporated, 6475 Christie Avenue, Emeryville, CA 94608, USA

INTRODUCTION..... 1

HOW TO USE THIS DOCUMENT 2

OVERVIEW OF PRODUCT DIFFERENCES 3

Structure..... 3

Administration 5

Language 12

Interfaces..... 16

Performance Differences..... 17

THE MIGRATION PROCESS 21

SUMMARY 29

APPENDIX A - PORTABILITY CHECKLIST: SERVER STRUCTURE 30

APPENDIX B – PORTABILITY CHECKLIST: T-SQL LANGUAGE 31

T-SQL Statement Differences..... 31

Option Setting Differences..... 34

Global Variable Differences 35

Keyword Differences..... 37

APPENDIX C - PORTABILITY CHECKLIST: DBA RELATED 38

Server Configuration Differences 38

Database Configuration Differences..... 40

DBA Command Differences..... 41

System Table Differences..... 46

Introduction

Microsoft SQL Server (MSSQL) and Sybase Adaptive Server Enterprise (ASE) are both SQL based client/server relational database management systems used to support information systems.

The two products have a common heritage because, until version 4.2, Microsoft simply licensed Sybase's database and so they were in fact two marketing identities for the same product¹. Since that point however, they have diverged. Microsoft have produced versions 6.0, 6.5 and 7.0 of Microsoft SQL Server, while Sybase have produced versions 4.8, 4.9, System 10 and System 11 of SQL Server and versions 11.5 and 11.9 of SQL Server's descendent, Adaptive Server Enterprise.

Users of Microsoft SQL Server may need to migrate applications to Adaptive Server Enterprise. Other customers may need to ensure that their applications are easily portable between the two products. Luckily, the unique history of these two products makes such conversion and portability relatively straightforward.

This white paper is primarily intended to assist with the application migration process from MSSQL to ASE. By migration, in this context, we mean the process of changing an application so that it uses Sybase Adaptive Server Enterprise, rather than Microsoft SQL Server, as its underlying database management system. It is also intended to be of assistance to those software developers needing to develop applications that can be easily migrated between the products.

The white paper presents an overview of the differences between the two products, from the point of view of an application developer or DBA, and then describes a systematic process for performing a migration from MSSQL to ASE. A series of checklists are also provided that should assist with migration planning and the design of applications that are portable between MSSQL and ASE.

¹ At that time, Sybase Adaptive Server Enterprise was known as Sybase SQL Server.

How To Use This Document

If you are reading this document, it is likely that you are in one of two situations. Either you have an existing Microsoft SQL Server application that you wish to migrate to Sybase Adaptive Server Enterprise, or you are considering building an application and wish to maximise its portability between the products.

For the first case, where there is an existing application, you should read the third and fourth sections of this document (*Overview of Product Differences* and *The Migration Process*). These sections will provide you with the information you need to migrate your application from Microsoft SQL Server to Adaptive Server Enterprise.

For the second case, where the aim is to maximise application portability, you should read the third section of this document (*Overview of Product Differences*) and then read the appendices which contain “*Portability Checklists*”. These sections will provide you with the information you need to maximise your application’s portability and so minimise any migration problems later.

This document presumes that you are using Microsoft SQL Server V7.0 and Adaptive Server Enterprise V11.9.2. However, most of the advice contained within it applies to earlier releases of the products too.

Overview of Product Differences

As you might expect of two products with so much shared heritage, ASE and MSSQL are similar database management systems. However, due to their separate development in later releases, there are quite a number of differences between them today. Luckily, most of these differences will not cause major problems for application portability between the products.

In this section, the differences between the two products are discussed. Not every difference between the two products is presented here, but all of the significant differences that need to be considered when porting an application are considered. The portability checklist, provided as an appendix to this document, lists additional low level details that may need to be considered.

The differences between ASE and MSSQL are presented as differences in server structure, differences in database administration, differences in the SQL language used, differences in application programming interfaces and performance related differences.

Structure

As might be expected, the structure of the two servers is extremely similar and any Microsoft SQL Server DBA will immediately find the structure of an ASE server familiar.

The most obvious difference between the two is the set of system databases. Both have `master`, `model` and `tempdb` databases that fulfil the same roles in both products. However, ASE stores its system stored procedures (such as `sp_role`) in a system database called `sybtempprocs`. An ASE server may also have system databases named `sybsyntax`, `dbccdb` and `sybtempdb` which are used for storing syntax information, DBCC utility output and distributed transaction management information respectively. Correspondingly, an ASE server does not have a `msdb` to hold system information, as an MSSQL server will.

Sybase and Microsoft have quite different philosophies when it comes to adding features to the database environment that are not core database facilities. Microsoft tend to add such features (such as backup and restore or monitoring) to the database server itself. Sybase tend to create new, dedicated, special purpose servers in order to offload this work from the database engine itself. For this reason, an ASE server will also normally have one or more “auxiliary servers” associated with it such as a backup server, a monitor server and an extended stored procedure server (XPS). An MSSQL server has no such associated auxiliary servers, so these are additional components a DBA will need to be aware of in the Adaptive Server Enterprise environment.

Similarly, replication in a Sybase Adaptive Server Enterprise environment is handled via the dedicated Sybase Replication Server product rather than directly by the ASE server.

Distributed transaction processing in the MSSQL environment is handled via Microsoft's Distributed Transaction Co-ordinator (DTC) which allows transactions to be co-ordinated between multiple MSSQL servers. DTC also allows MSSQL server to act as a resource manager within an XA distributed transaction. Sybase Adaptive Server Enterprise does not have a component directly equivalent to DTC. Distributed transactions between ASE servers are automatically co-ordinated by the transaction management component of ASE while integration into an XA transaction environment is achieved via the Sybase product XA-Server (an additional specialised server which is designed to be used in conjunction with Adaptive Server Enterprise).

Distributed query processing is offered by both environments. The Sybase Adaptive Server Enterprise produce offers a feature known as "*Component Integration Services*" ("CIS", available as a standalone product called OMNI). CIS allows objects in over twenty types of remote data source to be accessed as if they were local ASE database objects. The Microsoft SQL Server product offers the related feature, "*linked servers*". This feature allows access to objects in remote data sources that can be accessed via OLE-DB. The facilities offered by these features are rather similar, the main differences being the additional flexibility and transparency that CIS offers by the way it maps remote objects into the local namespace and the constraint of OLE-DB in the Microsoft solution.

The Microsoft SQL Server product provides some basic facilities to assist with configuration of "*fallback*" servers. These facilities provide system stored procedures that allow a database to be removed from one server and installed in another. These facilities are not available in Adaptive Server Enterprise.

In the area of server structure related to administration, there are two key aspects of MSSQL that need considered, the Job Scheduler and SQL Server Alerts. The Job Scheduler allows scripts or T-SQL batches to be scheduled for execution at some later time (either once or repetitively). SQL Server Alerts are a facility offered by Microsoft SQL Executive which constantly monitors the NT Event Log for MSSQL events and can perform some action in response to their occurrence. In the Adaptive Server Enterprise environment, both of these functions would be performed by user configured software outside the ASE server. For example, the operating system typically offers task scheduling services while alert handling can be achieved via a simple user written log scanner or by a more complex system management package supplied by a third party.

Finally, the other aspect of the two products at a server level is platform portability. ASE is available on a number of platforms (Solaris, HPUX, AIX, Dec Unix and Windows NT being the five primary operating systems supported). This means that ASE applications can be migrated between platforms with little or no change. MSSQL is only available for Windows NT and so is not suitable for applications where platform portability is a concern.

However, as you can surmise from the above points, the fundamental structure of the two products are really identical. This means that product structure is unlikely to cause any significant difficulty during application migration.

Administration

In this section, we consider the differences between the two products from the perspective of a DBA administering the server.

DBA Commands

Both products provide a set of DBA commands and a set of system stored procedures to allow management of the database server. The set of commands and stored procedures provided by the two products are very similar and DBAs will find it easy to move between the two products.

Some of the specific differences between the two products which a DBA will need to be aware of are:

- *Managing disk storage* – the two products share a rather similar abstract model of disk storage. However the details have diverged with the release of version 7.0 of MSSQL. Disk storage is defined to ASE as a series of “*disk devices*” (representing NTFS files or Windows NT disk partitions) which can then be used by one or more databases. This configuration is performed via the ‘`disk init`’ and ‘`create/alter database`’ commands. The MSSQL model removes this useful abstraction and talks directly about “*disk files*” which must be defined to the data server directly via the “`create/alter database`” commands. Hence, disk storage defined to MSSQL cannot be shared between databases. Similarly, management of “logical” areas of storage is performed via “*segments*” within ASE databases and via “*file groups*” within MSSQL. This implications of these differences are that storage management and database creation command scripts will differ between the two products. The other difference to be aware of is that MSSQL will silently increase the size of its disk file storage areas unless this is disabled when they are created. ASE will not increase the size of its disk devices.
- *Thresholds* – it is useful to be able to execute a stored procedure when a particular area of disk storage in a data server is becoming full. ASE provides this ability via its “*segment thresholds*” feature which allow a stored procedure to be executed when a storage segment fills to a certain point. MSSQL does not offer thresholds as such and this sort of processing must be performed via database alerts that must be created to occur when particular message numbers are logged by the database server. The alert must then be defined as a trigger for an appropriate database “*job*”.
- *Data base transaction log storage* – again, fundamentally both products are similar, with each database having its own “write ahead” transaction log. However, within ASE, the log is an integrated part of the database and can actually be read via a database table called “`syslogs`”. The DBA can make the decision as to whether the transaction log should be stored on separate disk devices from the database’s data. However as of version 7 of MSSQL, the transaction log is forcibly separated from the database into separate disk files and cannot be accessed via the “`syslogs`” table.

- *Disk mirroring* – DBAs must be aware that MSSQL does not offer integral disk mirroring and so if disk mirroring is required, it must be performed via the operating system.
- *The sp_configure stored procedure* – as with other sets of options within the database servers, both products offer the `sp_configure` stored procedure to allow server wide options to be specified. However, as discussed in a later section, the set of configuration options differs between the two products and so DBAs need to be aware of these differences.
- *The sp_dboption stored procedure* – like `sp_configure`, this stored procedure is offered by both products. However, as discussed in a later section, the set of options offered differs slightly. In addition, the ASE version of this stored procedure requires a checkpoint to be run in the database after all options for that database have been set.
- *The sp_tableoption stored procedure* - Microsoft SQL Server provides this stored procedure to allow tables to be marked as “*RAM resident*” or to be marked so that bulk loading will acquire table rather than row locks. Adaptive Server Enterprise does allow tables to be “pinned” in memory (but rather provides sophisticated memory management to avoid the problem altogether). Additionally, it entirely eliminates locking problems during bulk copy operations by only requiring locks for the end of the table for a short period at the end of the operation. Hence, this stored procedure is not required in the ASE environment.
- *Backup and Restore* – both products offer very similar `dump` and `load` commands for backup and restore of databases and transaction logs. However, Microsoft also provides commands called `backup` and `restore` for these functions and plan to remove the `dump` and `load` commands in a future version. Therefore, at this stage, very similar commands can be used with the two products. However, in the future, some changes to administration scripts may be required for the two products.

Some specific differences to be aware of between the two `dump` and `load` command implementations are that, the MSSQL commands require the type of media to be specified (tape, disk, floppy or pipe) while the ASE commands do not and will automatically sense the type of I/O device. The MSSQL “`load headeronly`” command is not available in ASE, this being achieved via “`load ... with headeronly`” (or `listonly`). The MSSQL commands refer to files on devices with multiple files by number while the ASE commands reference files by name. Finally, the MSSQL `stats`, `skip`, `noskip` and `expiredate` options are not available with the ASE commands.

- *The dbcc command* – both products offer the `dbcc` command for consistency checking of databases and other system level operations. However, due to its system level nature, the set of commands that it offers is slightly different between the two systems. While core sub-commands (like `checkalloc`) are the same, both systems offer their own sub-commands and DBA scripts that

use commands specific to one product (such as MSSQL's "dbcc shrinkdb") will need to be modified before they can be used with ASE. A full list of potential problems is provided as an appendix to this document.

- *Start-up procedures* – the Microsoft SQL Server product allows certain stored procedures to be marked as "start-up procedures" which the server will run automatically after it has started up. ASE does not offer a directly equivalent facility. Stored procedures that need run after ASE server start-up are usually executed via operating system facilities.
- *Auditing* - ASE has a comprehensive and powerful audit subsystem that allows audit trails of server activity to be captured at whatever level is appropriate for the user organisation. The audit subsystem has its own set of system stored procedures (e.g. sp_audit) to control it. Microsoft SQL Server really only provides auditing in a piecemeal fashion as a side effect of its performance monitoring features and an attribute of login events. Therefore, the DBA commands to control it are different (e.g. xp_loginconfig).
- *Open transaction monitoring* - Microsoft have added the opentran subcommand to dbcc to allow the oldest active transaction in each database to be identified. Within ASE, the master..syslogshold table can be used to provide the same information via a simpler mechanism
- *The nocheck option* – the MSSQL "alter table" command offers the nocheck clause which is used to skip checking of the existing data when adding "foreign key" and check constraints. This is the default behaviour within Adaptive Server Enterprise and so this clause is not needed and will create a syntax error if used in an ASE batch.
- *ANSI System Value Functions* - the MSSQL "create table" and "alter table" statements allow five of the ANSI standard's functions that return a system supplied value to be specified (CURRENT_TIMESTAMP, CURRENT_USER, SESSION_USER, SYSTEM_USER and USER). Adaptive Server Enterprise only allows the USER function to be used from this set (other system defined values can be inserted into columns by using defaults rather than system value functions).
- *Unique values* – both products offer the identity column attribute that indicates that the database should allocate a unique value for this column whenever a row is inserted. However, the Microsoft implementation allows a "seed" and "increment" to be specified at create time and a column may have the identity attribute if it is of type tinyint, smallint, int, decimal(p,0) or numeric(p,0). The Sybase implementation does not allow a seed or increment to be specified at create time (although an initial value can be placed in the table by the DBA). An ASE identity column must be of type numeric(p,0). Finally, the Microsoft identity column can be referred to by the pseudo name IDENTITYCOL whereas the Sybase version has the pseudo name SYB_IDENTITY. An application that makes use of identity columns may need to review their usage to ensure that it is possible to migrate

it. If it cannot be migrated directly, a custom counter using an integer column and an insert trigger can be used instead. The other thing to be aware of is that MSSQL 7 adds the `uniqueidentifier` datatype (to support Microsoft's extensive use of GUIDs in their system software). ASE does not support this datatype.

- *Server trace flags* – both products offer a number of “*trace flags*” which are used to alter the behaviour of the server in certain ways. Many of the commonly used trace flags have virtually identical behaviour between the two products (e.g. 302, 310, 3604 and 3605). Others (for example 1204) while not identical, cause similar information to be displayed. However as trace flags are really low level internal switches that often change from one release to another, it should not be assumed that any trace flags will work identically across both products. Sybase Customer Service and Support can advise on the use of particular trace flags.
- *Create “with encryption” clause* - Microsoft SQL Server offers the “with encryption” clause on the “`create procedure`”, “`create view`” and “`create trigger`” commands. This clause indicates that the associated query batch for the database object should be encrypted when it is held in the `syscomments` table. Adaptive Server Enterprise offers the same facility, but via the `sp_hidetext` system stored procedure. Hence, any “with encryption” clauses will need to be removed from `create` statements.
- *Create “for replication” clause* - Microsoft SQL Server has added the “for replication” clause to the “`create procedure`” statement to indicate that a stored procedure is being created for replication via MSSQL's replication feature. Replication in the Sybase environment is handled via the separate, purpose designed, server products Sybase Replication Server and Sybase SQL Remote, rather than by the Adaptive Server Enterprise server itself. Hence, there are no replication related extensions to ASE's T-SQL language and so this clause needs to be removed from any stored procedure creation batches.
- *Extended Stored Procedures* – while Adaptive Server Enterprise offers Extended Stored Procedures (ESPs) in a similar manner to Microsoft SQL Server, the set of system ESPs supplied is slightly different. In particular, the `xp_grantlogin`, `xp_loginconfig`, `xp_logininfo` and `xp_revokelogin` ESPs have been implemented as system stored procedures in ASE (called `sp_grantlogin`, `sp_loginconfig`, `sp_logininfo` and `sp_revokelogin`). Some Microsoft ESPs (e.g. `xp_msver` and `xp_sprintf`) are not available in Adaptive Server Enterprise.
- *Redefining Stored Procedures* – as of V7.0, MSSQL will allow a stored procedure, trigger or view to be updated “in place” via the “`alter proc`”, “`alter trigger`” and “`alter view`” statements. This isn't possible in ASE and so DBAs will simply need to use scripts to drop and re-create these

objects when they need changed (an approach which will help avoid errors and inconsistencies within an application).

- *Microsoft System Stored Procedures* – the MSSQL product adds a number of its own unique System Stored Procedures to the original set inherited from Sybase SQL Server version 4. Examples include `sp_altermessage`, `sp_dbremove`, `sp_helpsql` and `sp_makestartup`. In addition there are a number of system stored procedures (e.g. `sp_dboption`, `sp_serveroption`) that have slightly different usage between the two products. These differences may mean that some application and administration scripts may need to be updated to account for these differences. A list of the potentially problematic SSPs is supplied as an appendix to this document.

It will obviously be necessary to review existing administration scripts and change any use of these features. However, most of the updates are not difficult and so should not cause significant problems during migration.

Security

The two database servers offer very similar facilities in terms of security. In common with ASE, MSSQL provides security using *server logins*, system defined *login roles*, *database users*, *groups* of database users and a wide variety of *permissions* which can be assigned to users or groups.

All of these MSSQL security features are directly implemented in ASE and are used in exactly the same manner. In addition, both products offer the facility to use *trusted logins* when using authenticated connections (i.e. named pipes) on the Windows NT platform. This facility allows Windows NT users and groups to be assigned database permissions without needing a separate data server login.

In addition, both products offer the ability to create *user defined roles* which define groups of permissions that users can be granted authority to use. The roles can be organised in a hierarchy for efficient administration. The commands used in the two servers are different (e.g. the SQL92 “`create role`” command in ASE and the “`sp_addrole`” system stored procedure in MSSQL). One difference between roles in the two products is their scope. Roles in ASE are *server wide* to ensure consistency and easy administration. Roles in MSSQL are local to a *database* and so DBAs must take care to ensure that security is enforced consistently for multi-database applications. However, provided that this borne in mind, the security models can be mapped directly from one product to another for easy migration or porting.

In summary, the security that is provided by the database server should not cause problems during MSSQL to ASE migration as the two products offer almost identical facilities. The only real problems to watch for are role scope and command syntax.

DBA Procedures

The procedures that a DBA must follow are very similar for both products. In both cases routine tasks such as backing up databases, checking database consistency and

updating index statistics must be performed. Similarly, tasks such as adding logins and users, creating databases and database objects, tuning application queries and configuring software must also be performed.

As the two products are very similar, the details of most DBA procedures are in fact shared between them. Hence, the process of adding a server login is identical between the two products. Similarly, both products use the DBCC consistency checking utility to ensure internal database consistency (albeit often with different subcommands).

The differences between the two products really centre on the detailed usage of DBA commands and stored procedures and differences in the GUI utilities provided.

For example, when backing up a database to a disk file, the `dump database` command is used in both products. The difference is that the syntax for the MSSQL command is:

```
dump database db1 to disk="s:\dbdumps\db1.dmp"
```

while ASE's syntax for the same operation would be:

```
dump database db1 to "s:\dbdumps\db1.dmp"
```

Similarly, to force an extended stored procedure DLL to be unloaded from MSSQL, the `dbcc <dllname>(FREE)` command is used. The equivalent facility in the ASE environment provided by the system stored procedure `sp_freedll <dllname>`. Both commands are functionally equivalent and have similar usage, however the syntax needed differs between the two products.

As the sets of DBA commands and stored procedures are similar between the two products and the structure of the two products is almost identical, the migration of DBA procedures between MSSQL and ASE should be straightforward. It will simply involve DBAs becoming familiar with the differences of usage between the two products. A detailed list of differences to be aware of are listed in an appendix to this document.

As discussed below, the GUI utilities provided for DBAs by the two products are functionally similar but are obviously different in terms of usage. DBAs will therefore need to become familiar with the Sybase supplied GUIs (or use a third-party product which allows management of both types of database server simultaneously).

Configuration

In common with DBA procedures and commands, configuration of the two products is very similar.

In both cases, configuration is performed at the server and database level. Server level configuration is performed using the `sp_configure` system stored procedure while database level options are set using the `sp_dboption` system stored procedure.

Both products also provide a graphical configuration interface as part of their DBA GUI utilities.

Differences between the products in this area are concerned with the specific configuration options. Many options are identical between the two products, however

each offers many of its own unique configuration options that are not provided by the other.

There are a total of 42 server level configuration options in version 7.0 of Microsoft SQL Server (while ASE offers 136 as of version 11.9.2). Many of the MSSQL configuration options have analogous counterparts within ASE while a few are identical and others are not relevant to ASE. Specific information for the server configuration options is provided in an appendix to this document.

Similarly, Microsoft SQL Server offers 20 database level options in version 7.0 while version 11.5 of ASE offers 13 similar ones. Again, a number of these options are identical between the products while others can be mapped to similar options and a number can be ignored because they are not relevant.

Server and database configuration is unlikely to cause significant problems when performing MSSQL to ASE migration. Although there are differences between the configuration options offered by the two products, they are not serious and can be easily overcome during migration planning.

One aspect of configuration which can be problematic for applications needing to be portable between the products is that of character sets. ASE allows many character sets to be active in a server at once and so can easily support multi-lingual applications. However MSSQL only allows one character set to be active at once and so multi-lingual applications may be more problematic in this environment.

Data Import and Export

The data import and export facilities provided by the two products are almost identical. In both cases, binary database dumps can be created and loaded via the `dump` and `load` commands. In addition, the `bcp` (Bulk CoPy) utility is provided for extracting data from tables into binary or character data files.

Unfortunately, the binary files produced by `dump` cannot be used for migrating information between the two products. This is because the dump files produced by this process are image copies of database and transaction log pages that are inevitably different between the two products. Similarly, the “native” (i.e. binary) format files created by the `bcp` command should not be used to migrate data between MSSQL and ASE (particularly if they are on different platforms).

Migration of data from Microsoft SQL Server to Adaptive Server Enterprise may be achieved via:

- The “character” (i.e. text) format files produced by the `bcp` command (when used with the `-c` option). Note that the “-6” option should be used with MSSQL 7.0’s `bcp` utility to ensure correct formatting of dates.
- Replication technology (such as Sybase Replication Server) between the two products.

- Use of Adaptive Server Enterprise's Component Integration Services (CIS) component.
- Use of a third party product which allows whole databases to be unloaded in platform and release independent form and reloaded into another server.

These options are discussed in a later section of this document, "*The Migration Process*".

DBA Tools

There are a number of approaches that DBAs tend to use when administering database servers such as ASE and MSSQL. Some DBAs will prefer to use database commands and scripts to perform administration as this gives them the most power and flexibility. Others will prefer to use an administration utility with a graphical interface because of the simplicity and ease of use of this approach. In practice, most DBAs use both approaches.

Both products provide graphical DBA tools as standard (Microsoft providing *SQL Enterprise Manager* and Sybase providing *Sybase Central*). These graphical environments are broadly equivalent in terms of function although they are inevitably different in terms of specific usage and graphical display. In addition, a number of third party tools are available for both products (with some such as Embarcadero's *DB Artisan* being capable of managing ASE and MSSQL servers simultaneously).

Both products also provide a command line utility called `isql` that allows SQL scripts to be run against a server. The utility is very similar across the two products. MSSQL 7.0 also introduces a utility called `osql` (which is very similar to `isql` but uses ODBC to access the server). Obviously `isql` should be used in scripts which must be run against the two products.

Language

Typically, language differences account for most of the major problems when migrating applications from one relational database management system to another. However, in the case of Microsoft SQL Server and Sybase Adaptive Server Enterprise, the problems are massively reduced by the shared heritage of the two products.

Both database servers use Transact SQL (T-SQL) as their data manipulation, data definition and data control language. The two products use slightly different versions of this SQL dialect (which is based upon ANSI SQL 92) however the differences are minor when the overall language is considered.

Aspects of the T-SQL that can cause problems for migration or portability inevitably relate to Sybase and Microsoft specific extensions to SQL that have been added in later releases.

The primary T-SQL features, which may need to be reworked during a migration, include:

- *Datatypes* – there are a small number of datatype differences between the two products. Specifically Microsoft SQL Server’s character and text data types allow lengths up to 8000 characters while ASE’s enforce the SQL92 limit of 255 characters (use the `text` type to store longer strings). Data types found in MSSQL but not ASE are `cursor` (see later) and `uniqueidentifier` (use `identity` instead).
- *Triggers* - Microsoft SQL Server allows multiple triggers of the same type to be attached to a table (for example 2 delete triggers and 3 insert triggers). To achieve this in a more portable manner, place each of the trigger bodies in stored procedures and simply call these stored procedures from a single table trigger (creating temporary table copies of `inserted` and `deleted` if necessary).
- *CUBE and ROLLUP* – Microsoft has added two proprietary aggregate operators, `CUBE` and `ROLLUP` to T-SQL (`ROLLUP` is a subset of `CUBE`). These SQL extensions are not provided as part of ASE’s T-SQL language. These operators are used to produce “super aggregates” from a single “`GROUP BY`” statement. For example, when grouping by `col1` and `col2`, `CUBE` will produce aggregates for `col1`, `col2` and an overall group as well as the normal aggregates specified by the “`GROUP BY`”. Applications that use these operators will simply need to replace them with a number of “`SELECT ... GROUP BY`” statements for the aggregates that they actually need.
- *ANSI Join syntax* – The ANSI 92 SQL language definition added a verbose form of join syntax to the `SELECT` statement (using the keywords `JOIN`, `CROSS JOIN`, `INNER JOIN`, `LEFT OUTER JOIN`, `RIGHT OUTER JOIN`, `FULL OUTER JOIN`). Version 6.5 of Microsoft SQL Server added this syntax to its SQL dialect. ASE does not provide this syntax for the `SELECT` statement, but instead supports the traditional syntax using ‘=’, ‘*’ and ‘=*’. Microsoft SQL Server also supports this syntax so it should be used where migrations from Microsoft SQL Server to Adaptive Server Enterprise are likely.
- *SELECT lock syntax* – the MSSQL product allows the locking strategy for a `select` statement to be specified via the `NOLOCK`, `UPDLOCK`, `TABLOCK`, `PAGLOCK`, `TABLOCKX` qualifiers. ASE does not allow this syntax, leaving the locking strategy to the optimiser. Hence these qualifiers will need to be removed from any `select` statements that use them. An alternative in the ASE environment is the “`lock table`” statement which can explicitly request locks from a query batch.
- *SELECT TOP extension* – the MSSQL `SELECT` statement has been extended with a proprietary clause “`TOP`” which returns only a portion of the result set. This feature is available within the ASE environment via the “`set rowcount`” feature.
- *Rowsets* – the MSSQL `SELECT` statement has been extended to allow the use of pseudo-database objects called “row sets” (which are really OLE-DB

queries). Being based on proprietary Microsoft technology these aren't supported in ASE however a more open solution can be achieved using CIS to access remote data.

- *EXECUTE statement* – the Microsoft SQL Server `execute` statement has been extended to allow a character variable containing SQL to be executed (e.g. `exec "select * from t1"`). Adaptive Server Enterprise does not provide this directly, however with version 11.5 or later, the `sp_remotesql` feature of the Component Integration Services (CIS) facility (using the local server as a target) can be used to perform an identical function.
- *The BULK INSERT command* – version 7.0 of Microsoft SQL Server introduces this command which is simply a T-SQL version of the `bcp` utility. To access `bcp` from T-SQL within ASE, simply execute the utility command via `xp_cmdshell`.
- *Cursors* - as cursors were added to both databases in relatively recent releases, the details of their implementation in the two products are somewhat different. In particular, ASE implements cursors as defined by the SQL-92 standard. Microsoft have extended this in a number of proprietary ways and so as you would expect, there are a number of portability problems.

The primary features that cause problems are the “scrollable” features of the Microsoft cursor (e.g. moving backwards), cursor variables, global cursors and using cursors as parameters to stored procedures. These features are not available in ASE. In addition, the Microsoft cursor does not lock the underlying table while ASE ensures data integrity by retaining a shared lock on the current page. A less important difference is that cursor status is monitored via `@@sqlstatus` rather than `@@fetch_status` (with different status values).

A Microsoft SQL Server application that makes heavy use of scrollable cursors will need some rework to allow it to run well in the Adaptive Server Enterprise environment. This is particularly the case if the `prior`, `first`, `last`, `absolute` and `relative` keywords have been used on `fetch` statements. Similarly, applications that use cursor variables or parameters will need reworked. On the other hand, the result of the re-work will be an application that uses SQL92 features rather than proprietary extensions and which will probably exhibit much better performance in operation.

- *Optimiser hints* – both databases allow optimiser “hints” to be supplied as part of the `select`, `update`, `insert` and `delete` statements (e.g. “`select * from t1(<hint>)`”). However, the set of hints that may be supplied is different in the two products. MSSQL hints mainly concerned with the locking and join strategies to be used. The ASE hints include information about pre-fetch, I/O size and page caching (aspects of query optimisation not relevant to MSSQL's query processing). When migrating queries between databases, all hints should be removed and then, the queries re-tuned in the appropriate environment as required.

- *RAISERROR command* – the MSSQL `raiserror` command is somewhat different to the ASE one. Microsoft's was changed in V6.0 to be closer to a call to the C language `printf()` function. The Sybase `raiserror` is more SQL centric. Both have useful features (e.g. Microsoft's offers a severity field while Sybase's automatically converts argument types and allows for multiple languages). The translation from MSSQL's to ASE's is not difficult, but will probably take some effort during migration. One feature which must be addressed is the "with log" clause from MSSQL's `raiserror` statement (which indicates that a copy of the message should be placed in the server log). In the ASE environment, `with_log` specifier is only specified when the message is added to the server (via `sp_addmessage`) and so the "with log" clause can be removed from any `raiserror` statements. Similarly, the "with seterror" and "with nowait" clauses are not needed (they specify normal ASE behaviour) and can be removed.
- *SET statement* – the set statement is used to set session level options in both products. However, the set of options is slightly different between the two products. Microsoft SQL Server offers 38 session-level options (ASE offers 39). Again, some of these options are identical, some need mapped to equivalent options and others can simply be ignored .
- *The default keyword* - Microsoft SQL Server allows the keyword `default` to be used to indicate that a default value should be used when inserting data into a table or calling a stored procedure. ASE doesn't allow this syntax – defaults are simply used when no value is supplied.
- *The like operator* – the SQL `like` operator works almost identically between the two products with the exception of its handling of trailing spaces. The ASE `like` operator will ignore trailing spaces in the search pattern (i.e. "`%A%`" is the same as "`%A`") whereas the MSSQL `like` operator will not. If MSSQL language commands rely upon this to find trailing spaces, the search patterns will need reworked in ASE (use "[.]" for a trailing space).
- *System functions* - Microsoft SQL Server provides a number of system functions (such as `nullif()` and `stats_date()` which are not provided by ASE). During migration, these functions will need removed from any query batches that use them.
- *Global variables* – again, Microsoft SQL Server provides a number of global variables (e.g. `@@DBTS` and `@@SERVICENAME`) which are not available in Adaptive Server Enterprise. In total there are 8 global variables (from the total of 33 offered by MSSQL) that may cause problems.
- *Bound connections* - Microsoft SQL Server offers the concept of a "bound connection" where user transactions may be separated from a user connection and so multiple connections may participate in ("be bound to") a database transaction. ASE does not offer this facility at present and all activity for a user transaction must be completed using a single user connection. If a number of database clients must participate in a single transaction then the use of

transactional middleware (such as Sybase's Jaguar CTS or a TP Monitor) should be considered to achieve this.

- *The “begin distributed transaction” statement* – to start a transaction that involves several MSSQL servers, the `distributed` clause must be added to the “begin transaction” statement (this instructs the server to use DTC for transaction co-ordination). This clause is not necessary in an ASE environment and will be rejected with a syntax error (CIS will implicitly co-ordinate a distributed transaction between several servers).
- *Name resolution in stored procedures* – as of V7.0, Microsoft SQL Server uses “*deferred name resolution*” in stored procedures (meaning that objects referenced by a stored procedure do not need to exist when the stored procedure is created). In order to ensure run-time efficiency and integrity, ASE does not allow this. Therefore, objects referenced from stored procedures should be created before creating the stored procedure (in order to allow comprehensive checking at compilation time).
- *Keyword lists* – inevitably, the keyword lists for the two products are slightly different, reflecting the different extensions and priorities for the two products in later releases. A list of problematic keywords is provided in an appendix to this document.
- *Identifier lengths* – note that Microsoft SQL Server will allow up to 128 characters in an identifier (e.g. a table name). ASE (like other products, including Oracle) enforces the SQL92 maximum of 30 characters.
- *Table limits* - Microsoft SQL Server will allow up to 32 tables to be included in a single “SELECT” statement (i.e. a “join”). ASE will limit this to 16 tables. Similarly, MSSQL allows 1024 columns in a table whereas ASE will limit this to a more reasonable 250.

Although it appears that there are many differences between the T-SQL languages offered in the two products, in reality, the two dialects are far closer than is typically the case when comparing RDBMS languages.

The differences that there are between the products are differences in detail or particular vendor extensions which will be used rarely in an application. This means that migration of an application's SQL from Microsoft SQL Server to Adaptive Server Enterprise is significantly easier than is the case with most RDBMS migrations.

Interfaces

There are a number of programming interfaces that applications can use to access both types of database server.

Prior to version 7, the two primary application programming interfaces (APIs) used to access Microsoft SQL Server, were ODBC and DBLib. The ODBC API is used from many tools and languages, as it is a de-facto standard for database connectivity on

Microsoft operating systems (and is available on other platforms too). Microsoft specifically support DBLib as an API accessed from C and Visual Basic.

As of version 7, Microsoft are focusing on the use of their proprietary OLE-DB and ADO (Active/X Data Objects) COM interfaces as the primary interfaces used to access Microsoft SQL Server. This may cause major problems for customers who require portable applications that can be migrated easily across platforms.

Sybase Adaptive Server Enterprise is also accessible via the ODBC and DBLib APIs. In addition, ASE can be accessed via Sybase CTLib (Sybase's descendent API of DBLib) and OLE-DB.

The fact that ODBC provides portable database access means that ODBC clients should be able to run with minimal change against an ASE server as opposed to a MSSQL server. Provided that "pass through" mode has not been used to send incompatible T-SQL to the server, an ODBC client will simply require their ODBC driver to be changed. The Microsoft SQL Server driver is simply replaced with a Sybase Adaptive Server Enterprise driver.

Similarly, OLE-DB (and ADO) clients should be able to access both types of data server simply by changing the OLE-DB driver that they use. The problem with these interfaces is that they are only available to applications running on Microsoft's own operating systems.

An application that uses DBLib should simply be recompiled and linked using the Sybase DBLib libraries for the appropriate client platform. There are some parts of the Microsoft DBLib API that can cause migration problems (e.g., those parts concerned with the Microsoft Distributed Transaction Co-ordinator, which is not part of Sybase Adaptive Server Enterprise). However, it should be possible to migrate the majority of Microsoft DBLib software to use Sybase DBLib and ASE with minor changes. Customers requiring portable DBLib applications should be able to achieve this easily with a small amount of additional logic in their applications.

Presuming that a migration has been completed, it is worth considering the choice of client API to be used by the application in the future. Sybase recommend that new client development using 'C' or similar languages should use the CT-Lib API. CT-Lib is a new cross-platform client API from Sybase which was introduced with System 10 (and is not available in the Microsoft SQL Server environment). CT-Lib is a richer, simpler and more powerful API than DBLib and provides a number of advantages for application development and maintenance.

Performance Differences

The performance of a relational database system is a complex area which tends to be very specific to each product.

Some of the primary performance related differences between Sybase Adaptive Server Enterprise V11.9.2 and Microsoft SQL Server V7.0 which DBAs should be aware of are:

- *Query processor* – the two products today feature quite different query processors as both query processors have been extensively redeveloped since the products started to diverge. In particular, they feature different execution strategies, statistics and optimisation approaches. For example ASE allows fine DBA control over statistics and even allows direct update of statistics whereas MSSQL enforces a particular set of statistics that it attempts to collect automatically. On the other hand, they still have much in common at the functional level (for example, both use statistics based optimisation). DBAs will need to review the differences between them in order to ensure migrated applications perform well and to ensure application portability.
- *Locking* – once of the more vociferous debates about query processing in recent years has been the type of locking that database servers should use to protect data integrity. The core of the debate has centred around page locking versus row locking. Today, both products offer both locking schemes. However, ASE allows one of three locking schemes to be selected at the database object level (row level data only locking, page level data only locking or page level all locking). This provides a huge degree of flexibility for DBAs who need to optimise locking in their applications. Therefore, DBAs are recommended to review the locking strategy that they use in order to choose the best combination for their situation.
- *Process management* – the ASE product provides a “*logical process manager*” which allows workload to be managed within the database server (particularly for multi-processor servers). In particular, work can be partitioned up into groups with differing priorities and groups can be limited to use certain database server processors. Microsoft SQL Server does not provide such workload management and so applications needing to be portable may need to consider a more complex multi-server solution in the MSSQL environment.
- *Resource control* – both products offer some ability to control the amount of resource that any particular query can consume. MSSQL offers a simple “*query governor cost limit*” option which defines the maximum estimated elapsed time a query is allowed to use. ASE provides a fully featured Resource Governor which allows control over estimated or actual I/O amounts, actual row counts or elapsed execution time. In addition, the DBA can choose whether to enforce these limits prior to execution or during execution and can also specify the action to be taken when a resource limit is breached.
- *Memory management* – a key aspect of database performance is the management of memory in the database server as this has a profound effect on the amount of physical disk I/O that the server must perform. Microsoft SQL Server will attempt to dynamically resize its buffer cache to avoid operating system paging. This can obviously have a number of side effects (not least of which may be non-deterministic query performance). ASE will not attempt to change the amount of memory in use, leaving this crucial decision to the DBA. The key advantage that ASE offers in the area of memory management is that the buffer cache can be partitioned up into a number of smaller caches (called “*named caches*”). Each of these caches can be dedicated to a particular type of

workload (e.g. static lookup tables in one cache, OLTP tables in another, DSS in another etc.). In addition, the characteristics of each cache can be set differently to support the needs of the intended workload. Judicious use of named caches can produce dramatic improvements in server throughput due to the resulting reduction in interference between workloads.

- *Parallel execution* – another area of recent vigorous debate related to database processing has been that of parallel processing. There are two primary forms of parallel processing in a database server, inter-query parallelism and intra-query parallelism. Inter-query parallelism is by far the more important of the two and refers to the ability of the database server to execute many queries in parallel, switching between them as needed. Both products have supported this form of execution since the beginning. Intra-query parallelism refers to the ability of a database server to break a large query up into smaller sub-parts and execute each of these sub-parts in parallel. Although less important (only being of use for certain types of very large query) both products support this form of parallelism too.

Microsoft SQL Server provides a simple parallel facility that will automatically attempt to split up large SELECT statements and execute each part in parallel, when executing on multi-processor machines.

Sybase Adaptive Server Enterprise provides a more sophisticated parallel execution facility that provides the DBA with the ability to control the degree of parallelism used in various situations and at various levels. The optimiser also takes many factors into account when generating parallel execution plans (including physical data organisation and the current state of the server). This helps to ensure that parallelism is used appropriately and does not make matters worse on already busy servers or where underlying storage organisation does not lend itself to parallel execution.

- *Disk I/O* – the fundamental reason for having a database system is to store and retrieve data from disks. Therefore, the disk I/O processing performed by the database server is critical to achieving acceptable performance for large applications. Both products obviously attempt to maximise the speed of disk I/O in order to maximise server throughput.

Specific features Microsoft SQL Server offers in the area of disk I/O are “read ahead” processing and larger disk I/O units than previously. The “read ahead” processing is automatic (and not tuneable or controllable by a DBA). The database will attempt to read more data from disk that is immediately required, presuming that it will be used in the future. Similarly, the large disk I/O is rather simple, with the database server doing all I/O in units of 8Kb.

ASE offers a number of features in this area including variable I/O size (set at cache level rather than database level), asynchronous pre-fetch (again set at cache level) and table partitioning across disk devices. Careful use of these features along with named caches will result in a very highly tuned database

server that can achieve extremely efficient disk I/O even for servers supporting mixed workloads.

- *Monitoring* – in order to make use of the features outlined above, it must be possible to monitor the database server in order to establish where the performance problems lie. Microsoft SQL Server provides a system trace facility that can collect a large amount of raw data about server events. This information can be used to deduce performance problems with sufficient work. It also provides system stored procedures (like `sp_lock` and `sp_monitor`) and an integration into the Windows NT Performance Monitor. ASE offers the comprehensive monitoring system stored procedure `sp_sysmon` (as well as simpler tools like `sp_lock` and `sp_monitor`). It also provides an integration into the Windows NT Performance Monitor and a comprehensive server monitoring facility via Sybase Monitor Server (included with ASE) and the graphical monitor applets within the Sybase Central systems management tool.

Initially when performing migration or ensuring application portability, it is unlikely that these performance related areas will be of much concern. However, at some later point after initial development or migration, it is likely that some performance related work will be required. At this point, an understanding of the features offered by each database server will become important in order to allow performance and tuning work to be performed effectively.

The Migration Process

In this section of the document, a systematic process is presented for the migration of an application from Microsoft SQL Server to Sybase Adaptive Server Enterprise .

The result of following this migration process should be a completely migrated application running in an Adaptive Server Enterprise environment.

Step 1 Migrate the Server Structure

The aim of this step is to create an Adaptive Server Enterprise server structure that is capable of supporting the existing Microsoft SQL Server application in an ASE environment. Luckily, because the structure of the two products is so similar, this is unlikely to cause any significant problems when migrating.

Step 1.1 Create the ASE Server

An ASE server should be created to take the place of the MSSQL server. This is a straightforward step that can be performed using the Sybase graphical interface Server Config.

The structure of the server should mirror that of the existing MSSQL server (e.g. configuration of items such as the error log should be set to match the existing server for consistency).

Step 1.2 Configure the ASE Server

The server wide options for the ASE server should be set using `sp_configure`. These options should broadly match those of the existing MSSQL server (with any appropriate mapping for configuration settings that are different between the two products). See the appendix to this document for a list of MSSQL configuration parameters that may need reworked as part of the migration.

Step 1.3 Allocate Storage Space

Storage space for databases should be allocated in the new ASE server. This normally involves creating a script of “`disk init`” commands that are equivalent to the disk file statements used in the MSSQL server. If the original MSSQL commands no longer exist, they can be recreated using a tool such as *DB-Artisan* (or the devices simply recreated using DDL or a GUI such as *Sybase Central*).

Step 2 Migrate the Microsoft SQL Server Database

The aim of this step is to migrate the MSSQL databases to the ASE server. The result of this step should be an entire empty database structure that is ready to support the application.

Step 2.1 Create the Databases

The databases for the application can now be created (again, using the scripts from the MSSQL server if these are available, with the storage management clauses updated for ASE). As the two products have identical structures with respect to databases, the set of MSSQL server databases can simply be recreated in the new ASE server.

Step 2.2 Set the Database Options

The options for the newly created databases should now be set using `sp_dboption`. As discussed earlier, most of the database option settings can simply be copied from the MSSQL server. For those options that are not identical between the products, see the appendix to this document for suggestions on resolving mismatches.

Step 2.3 Create Server Logins for Database Owners

The server logins that own application databases should now be created. This server configuration can be taken directly from the security configuration of the existing MSSQL server and so is a very simple migration set.

Step 2.4 Set the Database Ownership

Finally, the database owner(s) for the application databases should be changed to the appropriate server login (again, this step can be copied directly from the existing server).

Step 2.5 Migrate Security Model

The migration of the security model is often a daunting task when migrating applications from one RDBMS product to another. However, because of the very similar security models offered by ASE and MSSQL, in this case it is fairly simple. All that is required at this stage is to confirm that the security model used by the application is appropriate and to make any modifications required to improve it or address the role scope differences between the products. With this complete, the logins, users, roles and permissions from the MSSQL server can simply be applied to the ASE server.

However, if moving from Windows NT to another operating system platform, it is important to consider the question of security integration with the operating system platform. Typically, users of Microsoft SQL Server use the security integration with Windows

NT. If migrating to ASE on Windows NT, the same operating system security integration is available. However if migrating to ASE on another platform (e.g. Unix) then this operating system security migration is not directly available and so a way of replacing it needs to be considered. One possibility for integrated security on Unix platforms is ASE's security integration with security principals such as DCE and CyberSafe.

Step 3 Migrate the Database DDL (Structure)

The purpose of this step is to create a database structure that can be populated with application data and then used by applications.

The output of this step should be the database with all database objects created, ready for data population.

Step 3.1 Obtain MSSQL Database DDL

The first step in the process of migrating the database is obviously to obtain the DDL used to create the database in the MSSQL server. If the original scripts are not available then a DBA tool (such as Microsoft *SQL Enterprise Manager* or Embarcadero *DB-Artisan*) or a CASE tool (such as Sybase *PowerDesigner*) can be used to recreate them.

Step 3.2 Create ASE Database DDL

The MSSQL DDL scripts should now be checked for Microsoft specific statements and options. A checklist to assist with this process may be found in an appendix to this document.

Where MSSQL specifics are found, these obviously need removed to allow the DDL to be used with the new ASE server.

The result of this step should be a set of DDL scripts that can be used with the ASE server to recreate the database structure.

Step 3.3 Run the DDL Against the Database

Finally, the updated DDL scripts should be used to recreate the database structure by running them against the new ASE server (typically as the 'sa' login).

Step 4 Review Administration and Security Procedures

In this step, the object level security and administration procedures for the application environment are migrated to the new ASE server. When this step is complete, the database should have adequate security for the application's needs and it should be possible to administrate the environment effectively. Therefore, the database server should be ready to accept application data and then be used by the application.

Step 4.1 **Migrate Security Scripts**

The overall security model for the application has already been migrated (step 2.5) in order to ensure that it was there when database objects were created. It is now important to review any security-related scripts that are part of the application (or any other aspect of the application that manipulates security-related objects).

Providing that there have been no changes to the application security model then the security scripts will require little or no migration work. Obviously, if the security model has undergone changes then security related scripts might require some related migration work.

Step 4.2 **Review Administration Procedures**

The existing administration procedures for the application now need to be reviewed. Firstly, they should be reviewed to ensure that they are optimal for the application and this is an ideal opportunity to improve any that have proved to be less than ideal in the past. Secondly, the procedures should be reviewed to ensure that they are appropriate to the application in the ASE environment.

In general, existing MSSQL administration procedures will be perfectly satisfactory in the ASE environment. However, it is possible that some MSSQL specific commands will need reworked.

One area that may need revision is backup and restore, if the MSSQL 7.0 specific `backup` and `restore` commands have been used (which will need replaced with `dump` and `load` commands).

The other area that may well need addressed is the use of the MSSQL scheduler. Should use be made of MSSQL's own scheduler, this will need changed during migration. The ASE product does not offer its own scheduling facility but rather relies upon the excellent schedulers available on the underlying operating system platforms. Any use of the MSSQL scheduler will simply need reworked to use an operating system hosted scheduler instead.

Step 4.3 **Migrate Administration Scripts**

Any existing application administration scripts should be reviewed at this point and MSSQL specific aspects of them replaced with ASE compatible commands and options. As most DBA commands and procedures are very similar between the two products, they are unlikely to cause major problems. However, minor changes (such as the syntax of the `dump` command for example) that may well be required could necessitate updates to administration scripts.

The result of completion of this overall step should be a complete ASE server with application databases ready to accept application data. All security should be in place

to protect the data as it is loaded and additionally, effective administration of the server should now be possible with the migrated administration procedures and scripts.

Step 5 Migrate Application Data

The data migration step of the process involves extracting data from the Microsoft SQL Server and then inserting that data into the newly created ASE server's databases.

Step 5.1 Export Microsoft SQL Server Data

The data in the Microsoft SQL Server needs to be extracted so that it can be migrated to the new databases in the Adaptive Server Enterprise server.

Possible ways of achieving this include:

- Using BCP to extract each table into a text file
- Using a desktop tool (such as Infomaker's *Data Pipeline*) to move the data from MSSQL into ASE
- Using ASE's CIS to "select" the data into the ASE databases
- Using Sybase Powerstage to smoothly migrate the data between systems
- Using a 3rd party product to extract the data

All of these methods will move the data fairly easily between the two servers. The method to choose depends upon the amount of data to be moved and the budget available for support tools.

The first two options are probably the simplest. Taking the first option, the `bcp` utility is supplied with both servers. To use it, simply extract each application table in the MSSQL server into a separate text file (use the `'-c'` and `'-6'` options to extract the data as ASCII characters compatible with ASE rather than binary data). This option has the advantage of being able to cope with huge amounts of data but some effort is required to prepare scripts to extract the data. The second option is easier (using a GUI to transfer the data in one step) but as all the data must travel via the desktop, this approach is limited in the amount of data it can handle.

The third option is more sophisticated and involves connecting the MSSQL server to the ASE via 11.5's *Component Integration Services* (CIS). The suggested approach is to create a "dummy" database (which requires almost no storage space) in the ASE server. This database can then be used to create "proxy" tables for the application tables in the MSSQL server (via the `"create existing table"` command). The data can then be moved from

these proxy tables to the real application tables using standard SQL (e.g. using “insert ... select”). The steps required for this process are explained in the manual, “*CIS Users Guide for ASE and OmniConnect*”. The use of SQL to perform the data transport also implies that any required data transformation can be performed during this step.

Finally, a dedicated product such as Sybase PowerStage (or another third party product) may be used to migrate the data. The cost and sophistication of this solution varies depending upon the product chosen, however there are products available which can migrate all of the application data with little or no user intervention.

Step 5.2 Import Data into ASE

In the case of some export approaches (e.g. using BCP) the import and export steps are separate. If this is the case, then the exported data obviously needs imported into the ASE server. In general, before performing the import, indexes, triggers and constraints should be removed from the application tables in the ASE server. This will allow the import access methods to execute as fast as possible. The constraints can be reapplied, the indexes can be rebuilt and the triggers recreated after the import is complete.

Step 5.3 Data Validation

With the data migration complete, the integrity of the data should be checked by data administration staff familiar with the application and its data. Ideally, standardised scripts will be available which perform logical data integrity checks with a minimum of administrator effort. Problems discovered at this stage should be checked against the source MSSQL server. If they are not found in the source server too, then the migration process may have failed in some way and needs investigated to address this.

The result of this step should be a populated application database with all of the application’s data correctly migrated and available in the ASE environment.

Step 6 Migrate Application Programs (Queries & Interfaces)

Up until this point, the migration process has focused on the database engine, the structure of the application database and the application’s data. This step moves on to migrate the application itself which involves ensuring that the application’s DML (i.e. SQL statements) will work against the new database and that the application programming interfaces are available against the new database server.

Step 6.1 Check Queries for MSSQL Specifics

The most fundamental part of migrating the application itself is to ensure that the queries that it runs will work correctly in the ASE environment.

There are two aspects to this, checking for invalid syntax to allow compilation and checking for semantic differences between the products to ensure correct execution.

All application SQL should be checked during this step. This primarily means T-SQL batches in application programs and scripts and application stored procedures. However, it is also important to check the application's triggers for correct compilation and execution.

Information to guide the checking process can be found in an earlier section of this document and additionally in an appendix.

The one major feature of Microsoft SQL Server which is likely to cause migration problems is scrollable cursors. This style of cursor is not available in Adaptive Server Enterprise and an application that uses them extensively may need significant re-design, re-code and re-testing to eliminate their use. The approach most likely to be successful with minimum effort is one which uses temporary tables to hold query results and then executes queries against the temporary table rather than using a scrollable cursor. If relative and absolute access to rows is required, an identity column can be added to the temporary table to act as a row number to allow rows to be identified by position.

Where Microsoft SQL Server specifics are found in queries, they obviously need to be removed to produce a version of the query that will compile and execute correctly in the ASE environment.

Step 6.2 Test All Application Queries

When the queries have been migrated, they should be tested to ensure that their execution in the ASE server is identical to their execution in the MSSQL server.

At the end of this step, all database resident SQL should be installed (i.e. stored procedures and triggers should be installed into the new ASE server's databases). This should mean that the entire database resident portion of the application has been successfully migrated to the ASE server.

Step 6.3 Rebuild Client Programs

With the application queries successfully migrated, the surrounding client programs can now be rebuilt if required. Obviously for programs written using 4GL environments like Visual Basic or PowerBuilder, little or no change to the client itself should be necessary provided that any MSSQL specifics were removed from their SQL batches in the previous step. For clients written in languages like 'C', it should be possible to recompile and link them using Sybase's DBLib as a replacement for Microsoft's DBLib.

Should client programs utilise specific Microsoft extensions to DBLib (such as DBLib's integration with Microsoft Distributed Transaction Co-ordinator) then these aspects of the programs will need to be removed or reworked.

Clients that use the ODBC, ADO or OLE-DB interfaces to access the database should run against the ASE server without change provided that MSSQL specific T-SQL batches have not been executed directly on the ASE server (e.g. a "SELECT . . . WITH CUBE" statement). The MSSQL driver(s) used with the MSSQL server should simply need to be replaced with suitable ASE driver(s).

Step 6.4 Test Client Programs

The client programs should now be ready for use by the application. All of the clients should now be tested to ensure that their behaviour is identical to those clients that were used with the MSSQL server.

With this step complete, the entire application has been migrated from Microsoft SQL Server to Sybase Adaptive Server Enterprise and is now ready for validation.

Step 7 Validate Migration

The final step in the migration process is to validate the migration with a thorough testing process. Ideally, there will be an existing automated set of integration, system and user acceptance tests that will allow easy validation of the migration. If such tests are not available, then an alternative testing process must be performed in order to ensure that the migration has been completed successfully.

The testing should include validation of:

- User interface transactions
- Batch processing
- Administration procedures
- Disaster recovery
- Application performance obtained

With all of this complete, the application can be considered to have successfully migrated from Microsoft SQL Server to Sybase Adaptive Server Enterprise .

Summary

This document has attempted to help you migrate your application or ensure its portability between Microsoft SQL Server and Sybase Adaptive Server Enterprise .

As you have seen, the two products are very similar. However, for the DBA and the developer, there are a number of differences which reflect differing priorities for the two products in later releases.

A migration process between the two is relatively easy and involves:

- Creating a suitable ASE server
- Migrating the application database structure
- Migrating the application's data
- Ensuring that administration and security procedures are appropriate for the new environment
- Checking that application SQL will work in the ASE environment
- Ensuring that programming interfaces will migrate correctly
- Testing the resulting migrated system to ensure compatibility with the original.

If these steps are followed, the migration of your Microsoft SQL Server application should be fairly straightforward and deliver real benefits quickly.

The process of ensuring portability is simpler than migration and fundamentally involves understanding the areas where the two products differ in order to minimise an application's dependency on such features. The checklists provided as appendices to this document should help to guide you during this process.

Appendix A - Portability Checklist: Server Structure

In this appendix to the document, a checklist is presented that attempts to the server structure related problems that are likely to be encountered when migrating applications between Microsoft SQL Server and Sybase Adaptive Server Enterprise.

Considerations for applications migrating from MSSQL to ASE are:

- ❑ Ensure that any user additions to the `msdb` database are migrated to another database in the ASE server.
- ❑ Ensure that any use of Microsoft SQL Server's fallback feature is removed from the application. (Note however that ASE can still be used with high availability software such as Microsoft Cluster Server, HP MC/ServiceGuard and IBM HACMP. See the Sybase Technical Information Library at: <http://techinfo.sybase.com/> for white papers explaining the configuration of ASE in these environments.)
- ❑ Ensure that any use of Microsoft Distributed Transaction Co-ordinator and/or bound transactions is reviewed with a view to replacing it with Sybase XA-Server and/or transactional middleware.
- ❑ Ensure that any use of Microsoft SQL Server's distributed query feature is replaced by use of ASE's CIS facility.

Additional considerations for ASE applications needing to stay compatible with MSSQL are:

- ❑ Ensure that there are no user additions in or application dependencies on the `sybssystemprocs`, `sybssystemdb` or `dbccdb` databases.
- ❑ If there are dependencies on CIS facilities for transparent access to remote data sources, a Microsoft SQL Server version of the application will need to use Sybase OmniConnect or Microsoft SQL Server distributed query facility to provide the remote data access.

Appendix B – Portability Checklist: T-SQL Language

In this appendix to the document, a checklist is presented that attempts to list all of the T-SQL language related problems that are likely to be encountered when porting applications between Microsoft SQL Server and Sybase Adaptive Server Enterprise.

Note that DBA related aspects of the T-SQL language that should be checked are listed in another appendix.

T-SQL Statement Differences

When migrating T-SQL to ASE from MSSQL the following points must be addressed:

- ❑ Remove the CUBE and ROLLUP aggregate operators from SELECT statements (replace them with multiple “ SELECT ...GROUP BY” operations).
- ❑ Replace any use of the “SELECT ... TOP” clause with use of “set rowcount”.
- ❑ Remove any verbose ANSI join syntax from SELECT statements (replace with =*, *= syntax).
- ❑ Replace any use of MSSQL’s OLE-DB “rowsets” by using CIS’s more transparent facilities.
- ❑ Check how T-SQL cursors are declared and remove any declared as “INSENSITIVE”. ASE only provides a “sensitive” cursor. To use an “insensitive” cursor in the ASE environment, create a temporary table (using “SELECT INTO”) to hold the query results and then declare a cursor on the temporary table.
- ❑ Check how T-SQL cursors are declared and remove any declared with the SCROLL qualifier as this option is not available in ASE. (See next point too).
- ❑ Rework any use of global cursors as ASE does not provide these objects.
- ❑ Check any use of the cursor fetch statement. The EXT, PRIOR, FIRST, LAST, ABSOLUTE and RELATIVE qualifiers are not available in the ASE environment (the only action allowed is to fetch the next row and this requires no qualifier). These qualifiers will need removed, however if an application relies upon this behaviour, this may cause significant problems during migration. It is suggested that an application that requires scrolling cursors instead uses an approach based upon creating temporary tables holding the cursor result rows and using discrete T-SQL SELECT statements on this table (or perhaps a cursor on this table). For identification of a particular row in the temporary table, an identity column can be used (so allowing “SELECT * FROM #restab WHERE rownum = 123” type queries to replace ABSOLUTE or RELATIVE scrollable cursors statements).
- ❑ Check for any use of “global” cursors which ASE does not support.

- ❑ Check any use of cursors and replace references to `@@fetch_status` with `@@sqlstatus`.
- ❑ Check for any use of the `uniqueidentifier` data type and the corresponding `rowguidcol` pseudo column (replace with an identity column, a binary column or an integer counter).
- ❑ Remove any use of the `cursor` data type. This will need reworked for use with ASE.
- ❑ Check for any use of `char`, `varchar`, `binary` or `varbinary` types where the length is greater than 255 (replace by use of `text` or `image` data type).
- ❑ Replace any “EXEC (“SELECT...”)” type statements with static T-SQL or use `sp_remotesql` to execute the dynamic batch.
- ❑ The `raiserror` command’s “with log”, “with nowait” and “with seterror” are not available in ASE (none of them are actually needed in the ASE environment).
- ❑ The `formaterror` command is not available in ASE (to access error message text, select information from the system catalogs).
- ❑ Remove the table, query and join optimiser hints (e.g. `INDEX`, `UPDLOCK`, `TABLOCK`, `HASH`, `MAXDOP`) from `SELECT` statements (allow the ASE optimiser to choose the optimal plan or use ASE’s hints).
- ❑ Avoid the `printf()` style of the Microsoft SQL Server `RAISERROR` statement (replace it with the simpler ASE style using implicit conversion and parameter numbering).
- ❑ Any use of the `DEFAULT` keyword on `INSERT` statements or stored procedure execution should be replaced by a syntax where the column or parameter is simply not supplied.
- ❑ Replace any use of the `NULLIF()` system function with the `CASE` statement which can be used for the same purpose.
- ❑ Replace any use of the `CAST()` function with `CONVERT()` (which is a portable synonym).
- ❑ The `GETANSINULL()` system function is not available in ASE. An alternative way to find the status for a database (e.g. called “db1”) is to use the query; “SELECT COUNT(1) FROM master..sysdatabases WHERE status&8192 = 8192 AND name=“db1””. This query will return 1 if the option is set and 0 otherwise.
- ❑ The `VAR()`, `VARP()`, `STDEV()` and `STDEVP()` aggregate functions are not available within ASE.

- ❑ The system functions `CURSOR_STATUS ()`, `DAY ()`, `MONTH ()`, `YEAR ()`, `APP_NAME ()`, `ISDATE ()`, `ISNUMERIC ()`, `NEWID ()`, `PERMISSIONS ()` and `PARSENAME ()` are not available within ASE.
- ❑ The system function `STATS_DATE ()` is not needed within ASE (simply query the system statistics tables directly).
- ❑ The meta-data functions `DATABASEPROPERTY ()`, `FILEPROPERTY ()` etc. are not available within ASE (query the system catalog instead).
- ❑ The string functions `UNICODE ()`, `QUOTENAME ()` and `REPLACE ()` are not available within ASE.
- ❑ The file object functions `FILE_ID ()`, `FILE_NAME ()`, `FILEGROUP_ID ()` and `FILEGROUP_NAME ()` are not relevant to ASE.
- ❑ Ensure that any user of the `LIKE` operator that involves trailing space in the search argument is reviewed (as ASE will ignore the trailing space – use the “ []” construct to represent a trailing space – e.g. “`LIKE 'ABC[][]'`” represents “ABC” followed by two spaces).
- ❑ If the application makes use of Microsoft SQL Server bound connections for multi-connection transactions, this aspect of the application will need to be revisited. As ASE does not allow transactions to be shared between connections, some form of transactional middleware (such as Jaguar CTS or a TP Monitor) must be used to achieve this.
- ❑ Remove any “`BEGIN DISTRIBUTED TRANSACTION`” statements, replacing them with “`BEGIN TRANSACTION`” statements (CIS will automatically co-ordinate multi-server transactions provided that this is possible for the data sources involved).
- ❑ Replace any “`set implicit_transaction on`” references with ASE’s “`set chained on`” syntax.

Additional considerations to ensure T-SQL statement compatibility between ASE and MSSQL are:

- ❑ The optimiser hints which ASE allows on `SELECT`, `UPDATE` and `DELETE` statements are not portable to MSSQL (e.g. the `LRU/MRU` clauses, the `PARALLEL` control hints and the `PREFETCH` hints).
- ❑ The ASE “`AT ISOLATION`” clause of the `SELECT` statement is not supported by MSSQL.
- ❑ Format strings used with `PRINT` and `RAISERROR` that have numbered parameters to allow for different languages are not portable to MSSQL.
- ❑ The “`ROLLBACK TRIGGER`” statement is not supported in MSSQL (a trigger must roll the whole calling batch back).

- ❑ The `ERROREXIT`, `PROCESSEXIT` and `MIRROREXIT` clauses of the `RAISERROR` command are not portable to MSSQL.
- ❑ The T-SQL data type conversion functions `inttohex()` and `hextoint()` are not available in MSSQL.
- ❑ The T-SQL string function `char_length()` is not available in MSSQL (use `datalength()` instead).
- ❑ The integrated security related functions `is_sec_service_on()` and `show_sec_services()` are not available in MSSQL.
- ❑ The role related functions `mut_excl_roles()`, `proc_role()`, `role_contain()`, `role_id()`, `role_name()` and `show_role()` are not available in MSSQL (MSSQL has two role related functions `is_member()` and `is_srvrolemember()` which may provide a partial solution for migration or application portability).
- ❑ The system functions `curunreservedpgs()`, `data_pgs()`, `lct_admin()`, `index_colorder()`, `ptn_data_pgs()`, `reserved_pgs()`, `rowcnt()`, `tsequal()`, `used_pgs()`, `valid_name()` and `valid_user()` are not available in MSSQL.
- ❑ Check that all identifiers are 30 characters long or less.
- ❑ Check that queries do not attempt to join more than 16 tables in one statement.
- ❑ Check that tables do not have more than 250 columns.

Option Setting Differences

When migrating T-SQL to ASE from MSSQL the following points must be addressed:

- ❑ The settings “`SET ANSI_NULL_DFLT_ON`” or “`SET ANSI_NULL_DFLT_OFF`” (which override the “`sp_dboption 'ANSI null default'`” setting) are not available in ASE. Similarly the “group” option `ANSI_DEFAULTS` and other ANSI option `ANSI_PADDING` are not needed in ASE.
- ❑ The MSSQL `ANSI_NULL` option is called `ANSINULL` in ASE.
- ❑ The MSSQL `CURSOR_CLOSE_ON_COMMIT` option is called “`close on endtran`” in ASE.
- ❑ Do not use the `DEADLOCKPRIORITY` option of the `SET` statement (this option isn't available in ASE).
- ❑ The MSSQL `FIPS_FLAGGER` option is called `FIPSFLAGGER` in ASE.
- ❑ The MSSQL `IMPLICIT_TRANSACTIONS` option is equivalent to ASE's `CHAINED` option.

- ❑ The MSSQL `LOCK_TIMEOUT` setting is equivalent to the ASE server “lock wait” option.
- ❑ The MSSQL `NUMERIC_ROUNDABORT` option is similar to the “arithabort numeric_truncation” setting in ASE.
- ❑ The `QUERY_GOVERNOR_COST_LIMITS` rather oddly allows an MSSQL session to set its own resource limits! Within ASE, the more conventional approach of the administration imposing resource limits on users is used! (Via the Resource Governor feature.)
- ❑ The `REMOTE_PROC_TRANSACTION` setting is equivalent to ASE’s `transactional_rpc` setting.
- ❑ The MSSQL `SHOWPLAN_TEXT` setting is equivalent to ASE’s `showplan` setting (there is no equivalent to MSSQL’s `SHOWPLAN_ALL` which produces a result set rather than a human readable plan).
- ❑ The MSSQL `XACT_ABORT` setting is not available in ASE (and so it should be left `OFF` for portable applications).

Additional considerations to ensure T-SQL option compatibility between ASE and MSSQL are:

- ❑ The standards related `ansi_permissions` setting is not available in MSSQL.
- ❑ The query processing settings `prefetch`, `process_limit_action`, “cursor rows”, “table count”, `char_convert`, “set statistics subquerycache” and `string_rtruncation` are not provided by MSSQL.
- ❑ The parallel processing settings `parallel_degree` and `scan_parallel_degree` are not relevant to MSSQL.
- ❑ The numerical processing settings “arithabort arith_overflow”, “arithabort numeric_truncation” and “arithignore arith_overflow” are not available in MSSQL.
- ❑ The security related settings `proxy`, `role` and “session authorisation” are not available in MSSQL.
- ❑ The CIS related setting `cis_rpc_handling` is not relevant to MSSQL.

Global Variable Differences

When migrating T-SQL to ASE from MSSQL the following points must be addressed:

- ❑ The `@@CURSOR_ROWS` global variable is not available in ASE (a “select count(1) from ...” statement can be used to find out how many rows would be returned by a cursor query).

- ❑ The @@DATEFIRST global variable is not available in ASE (the ASE server automatically changes date related constants in response to the user's language).
- ❑ The @@DBTS global variable is not available in ASE (the ASE server automatically inserts timestamp column values and does not provide a way to query any "current value" for this type).
- ❑ The status of a cursor should be checked using @@SQLSTATUS rather than @@FETCH_STATUS and the values in the variable are:

@@FETCH_STATUS=0	⇒	@@SQLSTATUS=0
@@FETCH_STATUS=-1	⇒	@@SQLSTATUS=2

The @@FETCH_STATUS of -2 cannot occur in the ASE environment as it only happens with scrollable cursors. In addition, @@SQLSTATUS can take the value 1 which indicates that the fetch statement resulted in an error (there is no equivalent value for @@FETCH_STATUS).
- ❑ There is no equivalent of @@MAX_PRECISION in ASE.
- ❑ There is no equivalent of @@LOCK_TIMEOUT in ASE (the server wide "lock wait period" setting can be extracted from master..sysconfigures if required).
- ❑ There is no equivalent of @@REMSERVER in ASE.
- ❑ There is no equivalent of @@SERVICENAME in ASE. The name of the Windows NT service for an ASE server is always "Sybase SQLServer _ NAME" and the internal service name is always "SYBSQL_NAME" where "NAME" is the name of the ASE server (e.g. "MYSQL").
- ❑ The @@PROCID variable works slightly differently in ASE. If a stored procedure causes a trigger to fire, the value of @@PROCID in the trigger is the ID of the stored procedure (so allowing the trigger to find out what caused it to fire). The Microsoft SQL Server version of @@PROCID will be set to the ID of the trigger in this case.

To ensure global variable compatibility between ASE and MSSQL, be aware that the following ASE global variables are not available in MSSQL:

- ❑ Query processing variables: @@IDENTITY, @@SQLSTATUS, @@TEXTCOLID, @@TEXTDBID.
- ❑ Parallel related variables: @@PARALLEL_DEGREE, @@SCAN_PARALLEL_DEGREE
- ❑ Transaction related variables: @@TRANCHAINED, @@ISOLATION, @@TRANSTATE.
- ❑ Language support variables: @@NCHARSIZE, @@CLIENT_CSID, @@CLIENT_CSNAME, @@MAXCHARLEN, @@CHAR_CONVERT.
- ❑ System variable: @@THRESH_HYSTERESIS.

Keyword Differences

Microsoft SQL Server keywords which are not keywords in the Adaptive Server Enterprise environment are: authorization, backup, bulk, committed, confirm, contains, containstable, controlrow, current_date, current_time, current_timestamp, current_user, deny, dummy, errlvl, errexit, file, filegroup, floppy, freetext, freetextable, identitycol, identity_start, insensitive, intersect, level, lineno, national, nocheck, nullif, numeric_transaction, openquery, openrowset, opendatasource, pipe, permanent, processexit, privileges, repeatable, replication, replace, restore, rowguidcol, scroll, serializable, session_user, some, system_user, tape, then, uncommitted, updatetext, when, work.

Adaptive Server Enterprise keywords which are not keywords in the Microsoft SQL Server environment are: activation, allpages, arith_overflow, at, authorization, cascade, char_convert, compact, connect, consumers, datapages, datarows, endtran, errordata, exclusive, exp_row_size, external, forwarded_rows, identity_start, lock, max_rows_per_page, membership, mirror, national, noholdlock, numeric_transaction, online, partition, passwd, privileges, proxy, readpast, readtext, rebuild, reclaim_space, reorg, replace, reservepagegap, resume, role, rows, session, share, shared, stripe, syb_identity, syb_restree, unpartition, user_option, using, work.

Any database objects created with names matching any of these keywords will need to be renamed during migration (probably by using an application prefix).

Appendix C - Portability Checklist: DBA Related

In this appendix to the document, a checklist is presented that attempts to the primary DBA related problems that are likely to be encountered when porting applications between Microsoft SQL Server to Sybase Adaptive Server Enterprise .

Server Configuration Differences

MSSQL server configuration specifics that need addressed during migration are:

- ❑ The “affinity mask” server configuration option does not have a direct equivalent in ASE (although there are some `dbcc` commands which can affect processor affinity on certain platforms - contact Sybase Customer Service and Support for more information if this is required).
- ❑ The "cost threshold for parallelism" option is not needed for ASE as its optimiser can make this decision on its own.
- ❑ The “cursor threshold” server configuration option is not needed in ASE.
- ❑ The "index create memory" configuration parameter is
- ❑ The “language in cache” configuration option is not needed by ASE.
- ❑ The "lightweight pooling" option is not needed by ASE (ASE will use fibres for user connections automatically).
- ❑ Note that MSSQL will attempt to dynamically change its "locks" configuration parameter depending upon server usage. ASE leaves this important memory related decision to the DBA.
- ❑ The “max async I/O” MSSQL configuration option is roughly equivalent to the ASE “max async i/os per engine” and “max async i/os per server” configuration options.
- ❑ The "max degree of parallelism" is equivalent to ASE's "max parallel degree" setting. (Note that ASE also provides the setting "max scan parallel degree" for further DBA control.)
- ❑ The "max query wait" setting is not available in the ASE environment.
- ❑ The "max server memory" and "min server memory" parameters act as bounds on MSSQL's memory usage. Again, rather than allowing the server to change its memory usage at will, ASE has a single "total memory" parameter used to set the memory size of the server.
- ❑ The "max text repl size" parameter is not relevant to ASE (replication is handled by the dedicated "Replication Server" product).

- ❑ The "max worker threads" MSSQL parameter is equivalent to ASE's "number of worker processes" parameter.
- ❑ The "media retention" configuration option is equivalent to ASE's "tape retention in days" parameter.
- ❑ The "min memory per query" MSSQL parameter is not needed in the ASE environment.
- ❑ MSSQL's "open objects" parameter is equivalent to ASE's "number of open objects" parameter (but note that again, ASE does not attempt to change this value itself, leaving this critical memory related decision to the DBA).
- ❑ The "priority boost" configuration parameter indicates that Microsoft SQL Server should run at a higher Windows NT priority level than normal. This option is equivalent to ASE's "-p" command line option which can be set via Sybase's graphical "Server Config" utility.
- ❑ The "query governor resource limit" is a server wide setting for MSSQL's rather simplistic resource governor. ASE provides a powerful, fully featured resource governor instead of this setting (see "Limiting Access to Server Resources" in the ASE Systems Administration Guide for more details).
- ❑ The "remote login timeout" and "remote query timeout" options are not available in ASE (it simply uses standard values).
- ❑ The "remote proc trans" option is equivalent to the "transactional_rpc" option in ASE.
- ❑ The "resource timeout" configuration option is not relevant to the operation of ASE.
- ❑ The "scan for startup procs" configuration option has no equivalent in the ASE environment.
- ❑ The "set working set size" configuration option is not necessary for ASE.
- ❑ The "show advanced option" configuration parameter controls how many of the configuration parameter sp_configure will display. A more flexible version of this function is provided by ASE's sp_displaylevel system stored procedure.
- ❑ The "spin counter" configuration option is not available in ASE.
- ❑ The "Unicode comparison style" and "Unicode locale id" are not necessary for ASE as UNICODE is simply another ASE character set.
- ❑ There is no ASE equivalent of the "user options" configuration option.

- ❑ The "VLM size" configuration option is not needed in the ASE environment (from 11.5.1 onwards, ASE can use large memory servers without further configuration).
- ❑ The `sp_serveroption` system stored procedure is rather different in MSSQL and ASE. In Microsoft SQL Server, this option is used to indicate a server's role in replication or fallback. In Adaptive Server Enterprise, this stored procedure is used to set inter-server network related options (for example whether inter-server password exchanges should be encrypted or not).

Additional ASE server configuration specifics that differ from MSSQL are:

- ❑ An ASE server has a text configuration file to hold its current configuration settings (conventionally called `%SYBASE%\SERVER.cfg` for an ASE server called "SERVER"). This can be used to share configuration settings between servers or to start servers with different pre-set configurations. This feature is not available with MSSQL servers.
- ❑ The ASE product has over 75 configuration settings that are not available in MSSQL. This is obviously too many to list meaningfully here, however DBAs who wish to configure MSSQL servers from ASE server settings should be aware of this.
- ❑ ASE's server wide memory management facilities (named caches and buffer pools) are not available in MSSQL and so the relevant system stored procedures do not exist.
- ❑ ASE also offers the ability to tune I/O size and disk I/O prefetch (via the named caches). MSSQL doesn't allow this configuration and so the corresponding configuration commands do not exist in MSSQL.
- ❑ There is no MSSQL equivalent of the Logical Process Manager and so there is no way of configuring priority queues for task types in MSSQL servers.
- ❑ Microsoft SQL Server does not offer a fully featured Resource Governor and the configuration stored procedures to allow various types of limits to be imposed on queries, users or applications do not exist in MSSQL.
- ❑ ASE offers cross-platform integrated security in the form of its Directory Services feature. This is not available in MSSQL and so there are no equivalent configuration stored procedures.
- ❑ Remember that MSSQL only allows a single character set to be loaded at once whereas ASE allows many. This may require configuration differences between products.

Database Configuration Differences

MSSQL database configuration specifics that need addressed during migration are:

- ❑ The "autoclose" and "autoshrink" MSSQL database options don't have equivalents in the ASE product.
- ❑ The "ANSI null default" MSSQL database option is equivalent to the ASE "allow nulls by default" database option.
- ❑ The "ANSI nulls" database option is not needed in ASE because of its transparent handling of UNICODE as a character set.
- ❑ The MSSQL "ANSI warnings" database option is equivalent to the ASE session settings for ANSI numeric compatibility.
- ❑ The "concat null yields null" database option has no equivalent in ASE.
- ❑ The MSSQL "cursor close on commit" database option is equivalent to the ASE session setting "close on endtran".
- ❑ The "default to local cursor" database option is not relevant to ASE.
- ❑ The MSSQL "offline" database option is not directly available in ASE (although "DBO use only" can usually be used to achieve the same effect).
- ❑ The "merge publish", "published" and "subscribed" MSSQL database options are not relevant to ASE.
- ❑ The "torn page detection" MSSQL database option are available in ASE

Additional ASE database configuration specifics that differ from MSSQL are:

- ❑ The ASE database setting "allow ddl in tran" is not supported by MSSQL as this is its default behaviour.
- ❑ The identity related database settings "auto identity", "identity in nonunique index" and "unique auto identity index" are not available in MSSQL databases.
- ❑ The ASE "abort tran on log full" and "no free space acctg" options are not supported by MSSQL.
- ❑ The ASE "allow nulls by default" database option is called "ANSI null default" in MSSQL.
- ❑ MSSQL does not allow space thresholds to be set for database segments and so DBAs who use ASE thresholds for executing log dump stored procedures or similar will need to rework this to use MSSQL alerts or similar facilities.

DBA Command Differences

The more important differences in DBA commands between the products are:

- ❑ The use of the `sp_configure` system stored procedure is slightly different between the two products.
- ❑ The use of the `sp_serveroption` system stored procedure is different between the two products.
- ❑ The usage of the `sp_dboption` system stored procedure is slightly different between the two products.
- ❑ The `sp_tableoption` Microsoft SQL Server stored procedure is not available in Adaptive Server Enterprise. To make a table RAM resident, a data cache can be created for it (with `sp_cacheconfig`) and the table bound to the cache (with `sp_bindcache`). This will prevent the table being flushed from memory by other server activity. The “`table lock on bulk load`” option is not relevant to Adaptive Server Enterprise because it does not need row or table locks during the bulk load process (a lock is needed only briefly on the last row or page at the end of the process).
- ❑ As discussed earlier, there are differences in the security commands to be used between the two servers, particularly related to roles. MSSQL system stored procedures such as `sp_setapprole`, `sp_addrole`, `sp_srvrolemember` do not exist in ASE and ASE’s SQL92 commands (such as “`create role`”, “`drop role`”) and related commands (like “`set proxy`”) will need to be used instead. Similarly, The Microsoft `xp_grantlogin`, `xp_loginconfig`, `xp_logininfo` and `xp_revokelogin` extended stored procedures are provided as system stored procedures in ASE (and so their names are “`sp_`” rather than “`xp_`”).
- ❑ As already explained, MSSQL 7.0 changes the storage model from “*devices*” to “*files*” and so disk storage related commands differ between the products.
- ❑ The replication related system stored procedures (`sp_addarticle`, `sp_droppublisher`, `sp_helpreplicationdb`etc.) are not provided by Adaptive Server Enterprise (the Replication Server product provides system stored procedures to perform similar tasks).
- ❑ The fallback related system stored procedures (`sp_fallback_enroll_svr_db`, `sp_fallback_permanent_svr`, `sp_fallback_withdraw_svr_db`etc.) are not portable to ASE.
- ❑ The “*target server*” related system stored procedures (e.g. `sp_msx_enlist`, `sp_post_msx_operation`) do not exist.
- ❑ The “*linked server*” related system stored procedures (e.g. `sp_addlinkedserver`) are not needed in ASE (use the CIS related stored procedures and commands instead).
- ❑ The agent/alert/job related stored procedures (`sp_add_alert`, `sp_help_jobstep`, `sp_apply_job_to_targets`etc.) are not portable to ASE.

- ❑ The start up processing related system stored procedures (`sp_makestartup`, `sp_unmakestartup` and `sp_helpstartup`) are not portable to ASE.
- ❑ The removable database related system stored procedures (`sp_create_removable`, `sp_dbinstall`, `sp_check_removeable` etc.) are not portable to ASE.
- ❑ It is not possible to attach and detach databases from ASE servers and so the related stored procedures (such as `sp_attach_db`) are not available in ASE.
- ❑ The web task related system stored procedures (`sp_runwebtask`, `sp_dropwebtask` etc.) are not portable to ASE.
- ❑ The Microsoft “full text” feature is not available within ASE (instead, ASE provides its own full text search speciality data store). Therefore, Microsoft’s `sp_fulltext_...` system stored procedures will need to be substituted for Sybase’s (see the “*Standard Full-Text Search Specialty Data Store User's Guide*” for details).
- ❑ The Microsoft `sp_OA...` system stored procedures are not available within ASE.
- ❑ It is not necessary to use complex system procedures like `xp_trace_...` to monitor ASE. Instead, simply use `sp_sysmon` and Monitor Server (with Sybase Central’s graphical monitor applets) or the Windows NT Performance Monitor.
- ❑ The `sp_change_users_login` system stored procedure is not available in ASE (although a DBA could perform the same function using an SQL batch).
- ❑ The `sp_coalesce_fragments` system stored procedure has no direct equivalent in ASE.
- ❑ The `sp_helplogins` system stored procedure is not available in ASE (although `sp_displaylogin` for a single login is and a DBA could add their own easily or install one from a public stored procedure library!).
- ❑ The `sp_setnetname` system stored procedure has no direct equivalent in ASE.
- ❑ Statistics related commands are somewhat different between the two products (ASE does not need to `sp_createstats` system stored procedure and does not offer automatic update via a procedure like `sp_autostats`). Also, the MSSQL “drop statistics” command is equivalent to ASE’s “delete statistics” command.
- ❑ The “load headeronly” command is not available in ASE (use “load ... with headeronly” or “load ... with listonly”). Similarly, the MSSQL specific backup and restore commands are not available within ASE.
- ❑ The “stats”, “skip”, “noskip” and “expiredate” options to the dump command are not available in ASE.

- ❑ The “disk”, “tape”, “floppy” and “pipe” qualifiers to the device named for the dump and load commands are not necessary in ASE.
- ❑ The “volume” qualifier to the dump and load commands is called “dumpvolume” in ASE.
- ❑ The “file” qualifier to ASE’s dump and load commands takes a file *name* rather than simply a number.
- ❑ The dbcc subcommands “inputbuffer”, “outputbuffer”, “showcontig”, “shrinkdatabase”, “shrinkfile”, “updateusage” and “useroptions” are not available in ASE’s dbcc command.
- ❑ The dbcc subcommand “show_statistics” is not needed within ASE (the optdiag command provides ready access to statistics).
- ❑ The dbcc subcommands pintable and unpintable are not needed in the ASE environment (use named caches instead).
- ❑ The dbcc subcommand opentran is not needed in the ASE environment (simply query master..syslogshold instead).
- ❑ The dbcc subcommand free is performed by the system stored procedure sp_freedll in the ASE environment.
- ❑ The dbcc subcommands perfmon and sqlperf are not needed in the ASE environment – use sp_sysmon instead.
- ❑ The validation dbcc subcommands checkfilegroup, checkident, newalloc, and textall are not provided (or necessary) within ASE.
- ❑ The dbcc subcommand proccache is roughly equivalent to ASE’s procbuf subcommand.
- ❑ The ASE server does not provide a direct means of running “startup” procedures (ASE customers usually do this via operating system means). Hence the system stored procedures sp_makestartup, sp_helpstartup and sp_unmakestartup are not available in ASE.
- ❑ The ASE “alter table” command does not have a “nocheck” option (because this is ASE’s default behaviour).
- ❑ The ANSI system information functions current_user, current_timestamp, session_user and system_user are not available in ASE (although the information is via the user_name(), suser_name() and getdate() functions).
- ❑ The set of trace flags provided by both products differs. Of the set documented by Microsoft, only 302, 310, 3604 and 3605 are officially supported by Sybase.

Contact Sybase Customer Service and Support for details of other trace flags similar to those supported by Microsoft.

- ❑ The MSSQL “with encryption” option to the “create procedure”, “create trigger” and “create view” command is not available in ASE. Use the `sp_hidetext` system stored procedure instead.
- ❑ The MSSQL “for replication” clause to the “create procedure” statement is not needed in ASE.
- ❑ Some of the Microsoft `sscanf` extended stored procedures (e.g. `xp_msver`, `xp_sprintf`, `xp_scanf`) are not provided in ASE (although a customer could write their own fairly easily).
- ❑ The Microsoft system stored procedures `sp_altermessage`, `sp_certifyremovable`, `sp_create_removable`, `sp_dbremove` are not available in Adaptive Server Enterprise.
- ❑ The “replace” option to the MSSQL `sp_addmessage` system stored procedure is not available in ASE – drop and recreate the user defined message instead.
- ❑ Stored procedures, views and triggers cannot be updated “in place” within ASE (however as the DBA should have recreation scripts to ensure application stability this should rarely be a major problem).

Additional ASE DBA command specifics that differ from MSSQL are:

- ❑ The ASE Logical Process Manager is not provided by MSSQL so configuration scripts that use the related stored procedures (such as `sp_bindexeclss`, `sp_addengine`, `sp_showpsexec` etc.) will need modified if they are to be used with MSSQL.
- ❑ The ASE Resource Governor is not provided by MSSQL so configuration scripts that use the related stored procedures (such as `sp_add_time_range`, `sp_add_resource_limit`, `sp_modify_time_range` etc.) will need modified if they are to be used with MSSQL.
- ❑ The CIS related commands and stored procedures (“connect to”, `disconnect`, “create existing table”, `sp_remsql` etc.) are not available in MSSQL.
- ❑ MSSQL does not offer partitioned tables so the “alter table ... partition” and “alter table unpartition” commands are not available. Similarly, the “update partition statistics” and “update all statistics” commands are not relevant to MSSQL.
- ❑ The “create table” statement’s “fill factor” clause is not portable to MSSQL.

- ❑ The “`max_rows_per_page`” qualifier of the “`create table`”, “`alter table`” and “`create index`” commands is not portable to MSSQL.
- ❑ The “`with override`” qualifier for “`create database`” and “`alter database`” is not portable to MSSQL.
- ❑ The “`create procedure ... as external name ...`” method of defining extended stored procedures is not available in MSSQL. The `sp_addextendedproc` system stored procedure (available in both products) must be used instead.
- ❑ The ASE `dbcc` subcommands `checkstorage`, `fix_text`, `indexalloc`, `reindex`, `tablealloc` and `tune` are not portable to MSSQL.
- ❑ The ASE storage and statistics management commands `reorg` and `optdiag` are not available within MSSQL.
- ❑ The `notify` option of the `dump` and `load` commands is not provided by the equivalent commands in MSSQL.
- ❑ The ASE “`online database`” command is not portable to MSSQL (as the MSSQL server will not take the database offline when load processing starts).
- ❑ There are quite a number of ASE administration related stored procedures that are not portable to MSSQL so administration scripts may need review to avoid problems with portability in this area.

System Table Differences

Some of the primary differences between the system catalogs in the two products are:

- ❑ The MSSQL `master..sysallocations` table does not appear in ASE (allocation information is held in `master..sysusages`).
- ❑ The `master..syslockinfo` table in MSSQL is functionally equivalent to ASE’s `master..syslocks` (although the details of the two tables are different).
- ❑ The MSSQL `master..sysoledbusers` and `master..sysperfinfo` tables have no direct equivalents in the ASE environment.
- ❑ The MSSQL `sysaltfiles`, `sysfilegroups` and `sysfiles` tables are not required in ASE as devices are held at server level. Information about devices in ASE is held in `master..sysdevices`.
- ❑ The MSSQL `sysforeignkeys` and `sysindexkeys` tables are equivalent to ASE’s `syskeys` table.
- ❑ The MSSQL `sysfulltextcatalogs` table does not exist in ASE.

- ❑ MSSQL splits security information into the `syspermissions` and `sysprotects` tables. In ASE, all of this information is held in the `sysprotects` table.
- ❑ The ASE system tables in the master database `sysengines`, `syslisteners`, `syslocks`, `sysloginroles`, `syslogshold`, `sysmonitors`, `sysremotelogins`, `sysresourcelimits`, `syssecmechs`, `sysrvroles`, `sysrangeranges`, `sysusages` are not found in the MSSQL master database.
- ❑ The ASE system tables `sysalternates`, `sysattributes`, `sysgams`, `syslogs`, `syspartitions`, `sysprocedures`, `sysroles`, `syssegments`, `systhresholds` and `sysusermessages` that will be found in all ASE databases are not present in the MSSQL system catalogs.