

MS SQL to Sybase® ASE Migration Guide

SYBASE, INC.
OCTOBER 2006

1 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. For various reasons, users of MSSQL may need to migrate their applications to ASE. 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.

The two products have a common heritage because, until version 4.2, Microsoft simply licensed Sybase's database server software; they were in fact two marketing entities for the same product¹. Since that point, however, they have diverged. Microsoft has produced versions 6.0, 6.5, 7.0, 2000 and 2005 of Microsoft SQL Server, while Sybase has produced versions 4.8, 4.9, System 10 and System 11 of SQL Server and versions 11.5, 11.9, 12, 12.5 and 15.0 of SQL Server's descendant, Adaptive Server Enterprise.

This document 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 ASE, rather than MSSQL, as its underlying database management system; this involves moving the schema and data from MSSQL to ASE as well as re-directing the application. 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 points of view of both the application developer and the DBA, and then describes a straightforward and systematic process for performing a migration from MSSQL to ASE.

The 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.

In the appendices a series of lists are provided that should assist with migration planning and application design so that future applications can remain portable between MSSQL and ASE.

This document attempts to provide a comprehensive overview of the differences between MSSQL and ASE and the migration issues to be faced when converting from MSSQL to ASE. However, it is not a complete list. Please consult the referenced ASE product manuals for complete syntax, sample usage, administration, and performance tuning issues. ASE product manuals are distributed with the ASE software on the Technical Library CD-ROM and can also be found online.

Some of the features for MSSQL and ASE mentioned in this document are bundled with the respective software products and some features are purchased separately as add-on options. Please consult with your Sybase sales representative for the list of bundled features and add-on options available for each version and packaging of ASE.

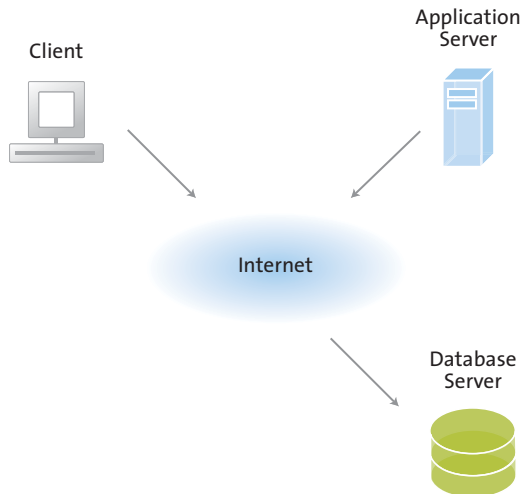
¹At that time, Sybase Adaptive Server Enterprise was known as Sybase SQL Server.

2 MIGRATION PROCESS

When migrating an MS SQL application all the application components have to be checked for specific Sybase issues. The application may contain MSSQL specific features. There are four main parts to migrating the application. The first part is migrating the data and SQL, the second part is migrating the applications that use the data, the third part is validating the migration and the final part is to optimize the performance of the migrated database.

2.1 Application Components

The architecture of an application to be migrated may look like:



The Client will communicate through the Internet with the Application server who will pass the messages to the Database server. All components used have to be compatible with the Sybase ASE server.

2.2 Migrating the Data

The migration of the data is the part of the migration project that can be partly automated. Tools can be used to create the schema, and move the data. The steps in data migration are:

- Creating the Sybase environment
- Creating the administration and security components
- Moving the data

The recommended tools for the migration are:

- PowerTransfer Tool to stream the data from MSSQL to ASE
- InfoMaker with its Data Pipeline feature to move the data from MSSQL directly into ASE
- MS DTS to move the data from MSSQL directly into ASE
- BCP for moving the data
- CIS (Sybase Component Integration Services) and Enterprise Connect Data Access (ECDA) to extract data from MSSQL and insert directly into ASE.

Appendix A describes the recommended datatype conversion and Appendix B contains the data schema details that the recommended tools cannot convert automatically.

2.3 Migrating the Application

The migration of the application consists of two parts. The SQL application code that resides in the database and the high-level language client application code, which accesses the database and executes SQL. This part of the migration will be a manual task.

Migrating the SQL application code consists of changing the MSSQL specific SQL. Stored procedures and triggers have to be checked for SQL language differences described in Appendix B. At this time also consider the tuning and performance considerations described in Appendix D.

There are five types of client application code that may need to be migrated. They consist of:

- Web application
- Embedded SQL application
- ODBC client application
- JDBC client application
- Database-specific library application

Like the SQL application code, part of the client application migration task is to check for SQL language differences described in Appendix B and tuning and performance considerations described in Appendix D. The other parts are discussed in Section 7 – Migrating Client Database Applications.

2.4 Validate Migration

The significant 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

2.5 Optimize the Performance

After the data and the application are migrated, check for the over all performance of the application. Derive performance of the application that is equivalent to or better than performance obtained by the SQL database. For deriving that consider the tuning and performance considerations described in Appendix D.

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

3 DATABASE ARCHITECTURE

3.1 SQL 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 MSSQL and ASE, the problems are substantially reduced by their shared heritage. 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 T-SQL that can cause problems for migration or portability relate to ASE and MSSQL-specific extensions that were added over time. Appendix B describes the SQL language differences that need to be addressed in the migration process.

3.2 Database and Data Storage

MSSQL and ASE servers are quite similar because much of the original architecture has been preserved. Both the server manage a collection of database; Both still use the original master, model, and tempdb system databases as well as many of the system procedures, database options, and configuration settings.

Both servers expect administrators to create additional user-defined databases containing application-specific tables, views, indexes, stored procedures, triggers, and other objects. Furthermore, both servers use a similar dialect of SQL (Transact-SQL), and support a similar protocol allowing client applications to connect and submit queries.

However, over time they have differed in how to add features beyond the core database services.

Microsoft tends to add features to the database server itself while Sybase usually creates new, dedicated, special-purpose servers to offload work from the database engine.

Sr. No.	Database Component	MSSQL	ASE
1	Backups	Integrated	Separated into Backup Server™
2	Monitoring	Integrated with Windows PerfMon and Event Manager	Separated into Monitor Server
3	Extended Stored Procedure	Integrated	Separated into XP Server
4	Replication	Integrated	Integrated with similar ASE Replication as well as separated into larger and more robust Replication Server®
5	Distributed Queries	Integrated with Linked Servers using OLE DB data sources for both homogenous and heterogeneous queries	Integrated with Component Integrated Services (CIS) for ASE-to-ASE queries and separated into Enterprise Connect Data Access (ECDA) for heterogeneous queries
6	Distributed Transactions	Integrated with MS Distributed Transactions Coordinator (DTC)	ASE provides a built-in Transaction coordinator, referred to as ASTC, for transparent two-phase commit coordination. ASE can also participate in distributed transactions managed by third party coordinators such as Tuxedo, Encina or DTC

Sr. No.	Database Component	MSSQL	ASE
7	Full-text Searching	Separated into Full-text Search Service	Separated into Full-text Search Engine
8	User-defined Functions	Integrated using T-SQL as the base language	Integrated using JAVA Virtual Machine with Java as the base language
9	XML Support	Integrated	Can be used as standalone or integrated into using ASE Java Support

Multiprocessor Support

Microsoft SQL Server and Sybase both support Symmetric Multiprocessor (SMP) configurations. MSSQL Server can take advantage of up to 32 processors on a single computer. MSSQL Server will distribute its tasks to threads running on the available processors. The number of processors to be use can be configured in MS SQL.

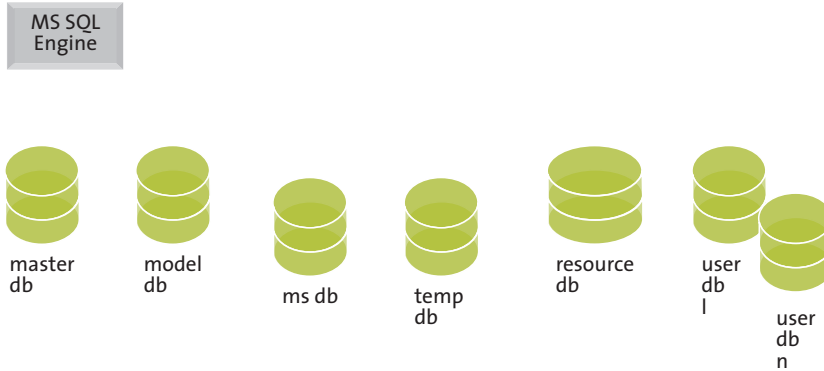
Sybase supports SMP using its Virtual Server Architecture (VSA). VSA comprises multiple database server processes (instead of threads), each running in a cooperative manner on a different processor using a shared everything architecture. Each database server process is referred to as an engine. Client requests are queued and are executed by whichever database engine becomes available to process the request. By default, Sybase is configured to use a single database engine. You can use the sp_configure "max online engines" command to add additional engines.

System Database

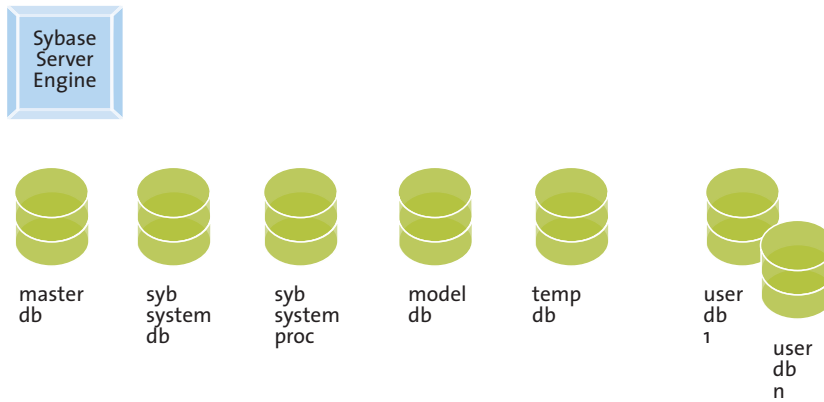
MSSQL and ASE each added system databases to the original set of master, model, and tempdb. In MSSQL 2005, system objects are not stored in the master database; instead they are stored in Resource database. Resource database is a read-only database that contains copies of all system objects that ship with MSSQL 2005. The ASE sybssystemprocs database holds the bulk of the system-stored procedures, which in MSSQL, reside in Resource database. MSSQL has an msdb database to store information for alerts, job scheduling, and a set of history tables for backup and restore. MSSQL may also have a distribution database when the server is configured for replication, which stores metadata and history data for all types of replication, and transactions for transactional replication.

The ASE sybssystemdb database tracks information about distributed transactions and manages two-phase commits. ASE also added a dbccdb database for holding information gathered by DBCC CHECKSTORAGE concerning database integrity faults and a sybsecurity database for its auditing subsystem. ASE servers can also have an optional sybsyntax database for storing syntax information retrievable through sp_syntax. In ASE, it is possible to define multiple user temporary databases (via CREATE TEMPORARY DATABASE) to reduce system table contention in tempdb when creating session specific temporary tables (e.g., #temp_table) and for database space reclamation of dropped temporary databases.

Overview of the main components of the MSSQL Server:



Overview of the main components of the Sybase ASE Server:



The optional databases can be installed optionally are:

Sybsecurity	Contains the audit system for Adaptive Server
Sybsystemdb	Stores information about distributed transactions
Sybmgmtdb	Stores jobs, schedules, scheduled jobs information, and data the internal Job Scheduler task needs for processing
<i>pubs2</i> and <i>pubs3</i> sample databases	Provided as a learning tool for Adaptive Server
Sybsyntax	Stores the syntax for T-SQL statements, system procedures, utilities and other routine for adaptive server

System Tables

The ASE and MSSQL master and user databases still share quite a lot of the original system tables. See Appendix C for comparison between the system views used by MSSQL Server and system tables of ASE.

Physical Data Storage

The storage systems used by Sybase and SQL Server databases differ significantly.

A MSSQL Server database comprises one or more physical database files, and one or more transaction log files. A database can have more than one Data files which can be grouped together in filegroups for allocation, administration purposes and placement purposes.

Sybase database and transaction logs are created on devices. A device is created using the Sybase DISK INIT command. This command pre-allocates space on a disk and associates it with a device name. Multiple Sybase databases can share the same Sybase device, and a single device can hold databases and transaction logs (although this is not recommended). As with SQL Server, the CREATE DATABASE, ALTER DATABASE, and DROP DATABASE commands are used to manage databases, although the syntax and options available are different. Extending a Sybase database requires allocating more space on a device, and it can require creating additional devices if all current devices are full (this is a manual process).

A database can consist of multiple segments each on their own logical device. A segment is a label that points to one or more database devices. Tables and indexes can be placed on different segments. The use of segments allows the developer to control where the data is placed in order to maximize I/O performance.

Logical Data Storage

The logical unit of data storage in both products is a data page, and a grouping of 8 pages is an extent. The MSSQL page size is fixed at 8KB. Objects are allocated one extent at object creation, but more than one object may reside on a mixed extent until they are able to occupy a full extent. The maximum size of a single row of non-image or text data is 8060 bytes (just under 8KB).

In ASE, a page size for the server can be one of 2KB, 4KB, 8KB, or 16KB and can only be specified when the server is created and therefore applies to all tables and indexes in the server. A larger page size raises the storage capacity of columns and rows up to the maximum of 16300 bytes (just under 16KB). Refer to limits for rows with variable and fixed-length columns for tables in the ASE Reference Manual for further details. Other ASE SQL and server structures (such as character expressions, variables and stored procedure arguments) have a maximum size of 16384 bytes no matter the page size. Each table and index is allocated space in units of one extent whether or not the extent is fully utilized.

Storage of Text and Image Data

MSSQL can store smaller text or image datatype data inline to the table using the sp_tableoption parameter "text in row"; otherwise, a pointer is stored to the separate data structure containing the text or image data much in the same manner as ASE. MSSQL can also target large data to specific files and filegroups with CREATE TABLE option TEXTIMAGE_ON. ASE stores text and image as a linked list of data pages separate from the table with pointers to data inline to the row. ASE can specify the target device segment for the large object by using sp_placeobject.

Both products offer table, page, and row-level lock granularities. MSSQL decides which locking strategy to use for tables and indexes dynamically. To override this feature, sp_indexoption is used to set whether a combination of page and table-level locking or row-level locking should be allowed or disallowed.

Locking

ASE allows the table owner to select one of three specific locking schemes for each table. The default-locking scheme is allpage locking, and this can be changed server wide. For table created with the datarows locking scheme, ASE locks only individual data rows within a data page (no index rows or pages are locked), which provides the highest degree of concurrency, but with the highest overhead. For a table created with the datapages locking scheme, ASE locks a data page only (no index rows or pages are locked). For the allpages locking scheme, ASE locks both data pages and index pages which provides the lowest degree of concurrency, but with the lowest overhead. In all three schemes, the server can decide to promote to table-level locking should the lock promotion thresholds set by the DBA be crossed. These locking mechanisms provide a wide degree of flexibility for DBAs who need to optimize concurrency, resource overhead, and space management in their applications.

3.3 Memory Management and Disk I/O

Both servers use a shared memory pool to maintain the state of server structures (such as user connections, locks, and active database object metadata), and to minimize physical disk I/O using a data buffer cache for data and index pages and a procedure cache for stored procedures, triggers, views, constraints, rules, defaults, and ad-hoc SQL batches. MSSQL dynamically grows and shrinks its data buffer cache to keep available physical memory on the machine to between “**max server memory**” and “**min server memory**”.

With ASE, the DBA sets the maximum amount of memory for the server to use with the configuration option “max memory”. The DBA configures the specific sizes and options of each of the data caches with `sp_cacheconfig`. The sizes of the procedure cache and server structures can be set and tuned dynamically, (i.e., without a database server restart) via `sp_configure`. Physical memory can be allocated dynamically as the structures are consumed by users (via the configuration option “dynamic allocation on demand”) or all at once at start-up. You can also allocate all of “max memory” at once (via the configuration option “allocate max shared memory”) to increase performance. Unlike MSSQL, ASE never dynamically reduces the amount of physical memory that it uses.

Two additional features of ASE memory management are named caches and large I/O buffer pools. Named caches allow the data buffer cache to be partitioned into multiple, smaller caches that 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.). The default data cache and other named caches can be further subdivided into buffer pools of varying I/O sizes of 1, 2, 4 or 8 pages via `sp_poolconfig`. A buffer pool with an I/O size that is eight times the server page size will read an entire extent in a single disk I/O operation. Judicious use of named caches and buffer pools can produce dramatic improvements in server throughput by reducing interference between competing workloads. For information on how to correlate the server configuration parameters for tuning memory, see Server Configuration Option Differences in Appendix E.

The disk I/O processing performed by the database server is critical to achieving acceptable performance for large applications. Both products attempt to maximize the speed of disk I/O in order to maximize server throughput.

MSSQL disk I/O is rather simple with the database server doing all I/O in units of one 8KB page. However, the MSSQL advanced scan feature allows multiple tasks to share full table scans by joining the tasks at the start of each request and then re-looping through the scan to provide missing rows to the late-comers.

Also, the MSSQL Storage Engine performs scatter-gather I/O allowing the transfer of data between cache and disk from non-contiguous areas of memory.

ASE I/O buffer pools, as discussed above, allow the disk I/O sizes to occur at units of 1, 2, 4, or 8 pages at a time. Depending on the server page size, disk I/O sizes range from 2KB to 16KB for a 2KB-page server up to 16KB to 128KB for a 16KB-page server.

MSSQL read-ahead processing allows more data to be read from disk than is immediately required, presuming that it will be used in the future—it is automatic and not a tunable component. ASE offers an analogous feature known asynchronous pre-fetch (APF); however the DBA may tune the maximum amount of memory in each of the I/O buffer pools to use for APF reads using `sp_poolconfig`. Careful use of ASE named caches, I/O buffer pools, and the APF feature will result in a very highly tuned database server that can achieve extremely efficient disk I/O even for servers supporting mixed workloads.

3.4 Database Security

Security in a database environment can be controlled at the network, server, and database levels. To access a database, a client may establish trust with the server through a security mechanism involving identification and authentication via a central authority such as a directory service and then through the database server itself. Once a client gains access to the database, database-level security features are applied.

The Security functions supported by Adaptive Server and MSSQL:

- **Identification and authentication** – Both MSSQL and Adaptive Server ensures that only authorized users can log into the system. In addition to password based login authentication, Adaptive Server supports external authentication using Kerberos, LDAP (Lightweight Directory Access Protocol), or PAM (Pluggable Authentication Module).

- **Security audit** – Both MSSQL and Adaptive Server includes an audit system. A comprehensive audit mechanism that checks access, authentication attempts, and administrator functions. The security audit records the date, time, responsible individual and other details describing the event in the audit trail.
An audit trail can be used to detect penetration of the system and misuse of resources. By examining the audit trail, a System Security Officer can inspect patterns of access to objects in databases and can monitor the activity of specific users. Audit records are traceable to specific users, which may act as a deterrent to users who are misusing the system.
In Adaptive Server, System Security Officer is the only user, who can start and stop auditing, set up auditing options, and processes the audit data. A System Security Officer, you can establish auditing for events such as:
 - Server-wide, security-relevant events
 - Creating, deleting, and modifying database objects
 - All actions by a particular user or all actions by users with a particular role active
 - Granting or revoking database access
 - Importing or exporting data
 - Logins and logouts
- **Discretionary Access Control** – Inside the database, logins and users form the basis for imposing further discretionary access controls in the form of roles and permissions. MSSQL forms logon identities from local and global group accounts from a Windows domain as the foundation of its database security. MSSQL organizes these login accounts using user-defined roles and grants them permissions with GRANT and REVOKE.
ASE retained the original concept of groups, which is a logical grouping of users in the database (for example, marketing, sales, etc.), and this is functionally similar to the MSSQL domain-defined groups. ASE DBAs can grant user-defined roles to logins and other roles.
- Both MSSQL and Adaptive Server implement the discretionary access control policy over applicable database objects: databases, tables, views, and stored procedures.
It provides access controls that give object owners the ability to restrict access to objects, usually with the grant and revoke commands. This type of control is dependent upon an object owner's discretion.
- **Division of Roles** – Both MSSQL and Adaptive Server allows an administrator to grant privileged roles to the specified users so only designated users can perform certain tasks. Both have the predefined system roles. Adaptive Server has predefined roles, called “system roles,” such as System Administrator and System Security Officer. In addition, Adaptive Server allows System Security Officers to define additional roles, called “user-defined roles.”
- **Protection of the TSF** – Adaptive Server protects itself by keeping its context separate from that of its users and by using operating system mechanisms to ensure that memory and files used by Adaptive Server have the appropriate access settings. Adaptive Server interacts with users through well-defined interfaces designed to ensure that its security policies are enforced.
- **Confidentiality of data** – Both MSSQL and Adaptive Server provides the data encryption for network protection. Adaptive server allows to maintain the confidentiality of data by encrypting client-server communications using the secure socket layer (SSL) standard or using Kerberos.
- **Resource utilization** – Adaptive Server provides resource limits to prevent queries and transactions from monopolizing server resources.
- **Password protected database dump and load** – ASE Protects database dump from unauthorized loads using the password parameter of the dump database command. If you include the password parameter when you make a database dump, you must also include this password when you load the database.

3.5 Process Management

ASE provides a Logical Process Manager (LPM) to allow the workload (particularly for multi-processor machines) to be managed within the server. In particular, work can be partitioned into groups with differing priorities, and groups can be limited to use certain ASE (CPU-based) engines. LPM allows work from an application, login, or stored procedure name to be granted special priority (high, medium or low) and affinity to specific CPUs (all or some) via `sp_addengine`, `sp_addexclass`, and `sp_bindexclass`.

MSSQL does not provide such extensive management and so applications needing to be portable may need to consider a more complex multi-server solution in the MSSQL environment. MSSQL will allocate threads from all instances to every CPU unless the "affinity mask" configuration option specifies which instance may not use a specific CPU.

3.6 Query Processor

ASE and MSSQL feature similar cost-based query optimizers using slightly different execution strategies, statistics, and optimization approaches. For example, ASE allows fine DBA control over table, index, and column histogram statistics, including allowing the direct update of the statistics. Most ASE table and index statistics are dynamically maintained while column distribution statistics must be manually updated using the `UPDATE STATISTICS` command. MSSQL enforces a particular set of statistics that it attempts to collect automatically.

DBAs will need to review the differences between the statistics in order to ensure migrated applications perform well and to ensure application portability.

3.7 Distributed Queries

MSSQL offers the linked servers feature to allow access to objects in remote OLE DB data sources via rowsets. ASE offers a feature known as Component Integration Services (CIS) to create proxy tables as references to tables located on remote servers. ASE and CIS allow for native distributed data across Sybase servers. CIS in conjunction with Sybase Enterprise Connect Data Access (ECDA) allows for remote access to objects in many types of remote data sources, including MSSQL, Oracle, DB2 (for OS/390, AS/400, Windows, and Unix), Informix and many ODBC data sources.

3.8 Distributed Transactions

Distributed transactions in MSSQL are handled via Microsoft's Distributed Transaction Coordinator (MS DTC) allowing transactions to be coordinated between multiple MSSQL servers and other database servers. MS DTC also allows MSSQL server to act as a resource manager within an XA distributed transaction.

MSSQL stores the outcome of the transactions in `msdb`. For ASE, distributed transactions may be coordinated natively or through an external transaction manager. The built-in Adaptive Server Transaction Coordinator (ASTC) automatically coordinates transactions distributed via CIS allowing transparent two-phase commit operations between ASE and Oracle, while providing best-effort coordination between other third-party databases (such as MSSQL, DB2, Informix) using Sybase's ECDA servers.

The ASE Distributed Transaction Manager (DTM) communicates with third party transaction managers (such as CICS, Tuxedo, Encina, or MS DTC) to participate in both XA and MSDTC distributed transactions. ASE stores the outcome of distributed transactions in `sybssystemdb`. See the ASE product manual Using Adaptive Server Distributed Transaction Management Features for more information.

3.9 Transaction Processing

Adaptive Server automatically manages all data modification commands, including single-step change requests, as transactions. By default, no transaction is started without the “begin transaction” statement paired with **commit transaction** or **rollback transaction** statements to complete the transaction. This can be changed by the set option (set chained on). All transactions are stored in the Sybase transaction log. Each database has its own transaction log. This should be large enough to hold all current transactions for active users and should be truncated or backed up on a regular basis.

A transaction can be cancelled or roll back with the **rollback transaction** command any time before **commit transaction** has been given. Using savepoints, either an entire transaction or part of it can be cancelled. However, a transaction cannot be cancelled after it has been committed.

To support SQL-standards-compliant transactions, Adaptive Server allows you to select the mode and isolation level for your transactions. Applications that require SQL-standards-compliant transactions should set those options at the beginning of every session.

The global variable @@transtate keeps track of the current state of a transaction. Adaptive Server determines what state to return by keeping track of any transaction changes after a statement executes. Adaptive Server does not clear @@transtate after every statement. It changes @@transtate only in response to an action taken by a transaction.

Transaction mode of the procedure can be set by system stored procedure sp_procmode. For example:

```
sp_procmode procedure_name, "chained"
```

To run the stored procedure under either chained or unchained transaction mode,

```
sp_procmode procedure_name, "anymode"
```

Certain data definition language commands in transactions can be used by setting the ‘ddl in tran’ database option to true. If ‘ddl in tran’ is true in a particular database, commands such as ‘create table’, ‘grant’, and alter table can be issued inside the transactions in that database. To check the current settings of ‘ddl in tran’, use command sp_helpdb.

Isolation Level:

By default, the Adaptive Server transaction isolation level is 1. The ANSI SQL standard requires that level 3 be the default isolation for all transactions. This prevents dirty reads, nonrepeatable reads, and phantom rows. To enforce this default level of isolation, Transact-SQL provides the transaction isolation level 3 option of the set statement. This option instructs Adaptive Server to apply a holdlock to all select operations in a transaction. For example:

```
set transaction isolation level 3
```

Applications that use transaction isolation level 3 should set that isolation level at the beginning of each session. However, setting transaction isolation level 3 causes Adaptive Server to hold any read locks for the duration of the transaction. If you also use the chained transaction mode, that isolation level remains in effect for any data retrieval or modification statement that implicitly begins a transaction. In both cases, this can lead to concurrency problems for some applications, since more locks may be held for longer periods of time.

To return your session to the Adaptive Server default isolation level:

```
set transaction isolation level 1
```

3.10 Parallel Execution

Intra-query parallelism refers to the ability of a database server to break a large query into smaller subparts and execute each of these sub-parts concurrently. Both products support this form of parallelism. MSSQL provides a simple parallel query facility that will automatically attempt to split up large SELECT statements and execute each part in parallel when executing on multi-processor machines.

ASE provides a more sophisticated parallel execution facility using worker processes that provide the DBA with the ability to control the degree of parallelism used in various access modes (partition-based or hash based scans) and in various situations (single-table accesses, joins, sorts, index creation, etc.). The optimizer also considers many factors when generating parallel execution plans (including physical data organization and the current server resources) to help to ensure that parallelism is used appropriately and does not make matters worse on already busy servers or where underlying storage organization does not lend itself to parallel execution.

3.11 Platform Portability

MSSQL is only available for Windows operating systems. ASE is available on a number of platforms in addition to Windows, including Intel Linux, Sun Solaris, Hewlett Packard HP-UX, IBM AIX, Compaq Tru64 Unix, Silicon Graphics IRIX, and Mac OS X. The larger platform selection means that ASE applications can be migrated between hardware platforms with little or no change.

3.12 Backup and Restore

Both products offer very similar commands for backing up and restoring databases and transaction logs. MSSQL currently favors the commands BACKUP and RESTORE while ASE uses DUMP and LOAD. In actuality, MSSQL provides synonyms for its older DUMP and LOAD commands to point to the newer BACKUP and RESTORE, but there are plans to remove the older commands in a future version.

MSSQL backups can be used to selectively restore damaged files or filegroups without restoring all the files in a database. MSSQL provides for striping of database files across a maximum of 64 backup devices.

ASE backups are performed by the Backup Server to offload processing from the ASE server itself. Backup Server dumps an entire database image (i.e., all the used pages in the database) to up to 512 simultaneous dump devices (tapes or files), and that image must be restored in its entirety (that is, no partial loads).

MSSQL transaction logs are backed up and restored in the same manner as ASE and both have analogous recovery models. The MSSQL "simple" recovery model (set with ALTER DATABASE) truncates the transaction log when a checkpoint occurs and is analogous to the ASE database option "trunc log on chkpt". Both products allow transaction logs to be restored to a specified point in time, though MSSQL also has recovery to a specific point of work via a marked or named transaction.

Through the MSSQL differential backup option, only the pages that have changed since the last backup are recorded. ASE does not provide this feature, however ASE does have an option for dump file compression that can reduce the dump file size by upwards of 80% and which can substantially reduce the time to perform a backup. In addition, ASE provides the QUIESCE DATABASE command to suspend update activity for short periods of time where a high-speed disk copy program can make backups of the database devices. Third party vendors provide these programs in support of 24x7 operations of VLDB databases.

The MSSQL backup commands require the type of media to be specified (e.g., tape, disk, or named pipe) unless a logical dump device is used. The ASE Backup Server will automatically sense the type of device (tape or disk) and will act accordingly. For more information on option differences between the BACKUP and DUMP commands, see DBA Command Differences in Appendix B.

Automatic recovery of ASE after a system failure or shutdown

Each time you restart Adaptive Server—for example, after a power failure, an operating system failure, or the use of the **shutdown** command—it automatically performs a set of recovery procedures on each database.

The recovery mechanism compares each database to its transaction log. If the log record for a particular change is more recent than the data page, the recovery mechanism reapplies the change from the transaction log. If a transaction was ongoing at the time of the failure, the recovery mechanism reverses all changes that were made by the transaction.

On start of Adaptive Server, it performs database recovery in this order:

- Recovers master.
- Recovers sybssystemprocs.
- Recovers model.
- Creates tempdb (by copying model).
- Recovers sybssystemdb.
- Recovers sybsecurity.
- Recovers user databases, in order by sysdatabases.dbid, or according to the order specified by **sp_dbrecovery_order**. See below for more information about **sp_dbrecovery_order**.

Users can log in to Adaptive Server as soon as the system databases have been recovered, but they cannot access other databases until they have been recovered.

3.13 Basic Tuning

These are the basic Sybase ASE server tuning considerations:

- Sizing tempdb, “tempdb” database, which is a temporary database area used for query processing worktables and temporary user tables. Tempdb will be created using a template, the model database, and will be, by default, 2 MB. In almost all cases, tempdb’s size should be increased to handle any worktables or temporary user tables required by the application(s).
- sp_configure “total memory”. Always give ASE as much memory as is available on the server machine. As a rule of thumb, for Solaris, and if the server machine is dedicated to ASE, 85% of the physical memory may be given to ASE. For a dedicated HP server, 75% is the rule. However, the best way to estimate how much memory is available is to account for all the processes (including the ASE processes that will run) and find out what memory is available.
- sp_configure “number of user connections”. The default value for this option is “25”. In most cases, this will not be enough to support all user sessions and processes on ASE. Account for all users, dump/load, replication server and other sessions that use connections.
- sp_configure “lock scheme”. By default, ASE’s locking scheme when creating tables is “allpages”. This means that locks are taken on data and index pages and if you are considering using “datapages” or “datarows”, you may set the option server-wide using the sp_configure option. Remember that changing this option will not automatically convert any existing tables.
- sp_configure “number of devices”. The default value is “10”. This value should be increased to support the number of database devices you will be creating using disk init.
- sp_configure “procedure cache percent”. The default for this option is “20” (20%). If your applications make heavy use of dynamic SQL or stored procedures, consider increasing this percentage. The procedure cache, like most of the configurable options, may be monitored using sp_sysmon and the procedure cache sub-report to help established if it is sized correctly (see Performance and Tuning guide, chapter 24, “Monitoring performance using sp_sysmon”).
- sp_configure “number of open objects”, The total should account for the maximum number of objects used/accessed during application processing and include tables, indexes, triggers, constraints, rules, stored procedures and dynamic SQL. In some cases, you may want to set the value higher temporarily to support set up activities, for example data migration.
- User-defined caches and cache partitioning, ASE allows you to partition the default data cache by creating user-defined caches and/or partitioning, using sp_cacheconfig or sp_configure “local cache partition number”. This is very useful in an SMP environment where contention can be experienced by multiple processes trying to write to default data cache at the same time (see Performance and Tuning guide, Chapter 10: Memory Use and Performance).
- sp_configure “max network packet size”, the maximum packet size can be decided, when a lot of data is send the network overhead can be reduced by setting a larger size.
- sp_configure “number of worker processes”, this will allow certain queries to be processes more optimally.

- `sp_configure` "max online engines", this will allow the workload of the server to be spread over more engines. The max number of engines should not exceed the number of CPU's available on the machine.
- `sp_configure` "tcp no delay", set to enable will make sure the packets are send immediately.
- `sp_configure` "optimization timeout limit", this will specify the amount of time Adaptive Server can spend optimizing a query as a percentage of the total time spent processing the query.
- `sp_configure` "statement cache size", *size_of_cache*

The statement cache allows Adaptive Server to store the text of ad hoc SQL statements. Adaptive Server compares a newly received ad hoc SQL statement to cached SQL statements and, if a match is found, uses the plan cached from the initial execution. In this way, Adaptive Server does not have to recompile SQL statements for which it already has a plan.

The statement cache is a server-wide resource, which allocates and consumes memory from the procedure cache memory pool. Set the size of the statement cache dynamically using the statement cache size configuration parameter.

The *size_of_cache* is the size, in 2K pages:

For example, to set your statement cache to 5000 2K pages, enter:

```
sp_configure "statement cache size", 5000
```

- The convenient way of matching query demands with the best optimization techniques is optimization goals. It ensures optimal use of the optimizer's time and resources. The query optimizer allows you to configure two types of optimization goals, which you can specify at three tiers: server level, session level, and query level. Set the optimization goal at the desired level. The server-level optimization goal is overridden at the session level, which is overridden at the query level. These optimization goals allow you to choose an optimization strategy that best fits your query environment:

allrows_mix – the default goal, and the most useful goal in a mixed-query environment. It balances the needs of OLTP and DSS query environments.

allrows_dss – the most useful goal for operational DSS queries of medium to high complexity. At the server level, use `sp_configure`. For example:

```
sp_configure "optimization goal", 0, "allrows_mix"
```

At the session level, use **set plan optgoal**. For example:

```
set plan optgoal allrows_dss
```

At the query level, use the **select** or other DML command. For example:

```
select * from A order by A.a plan
"(use optgoal allrows_dss)"
```

3.14 Other Administrator Tasks

Sybase ASE provides a number of Administrator tools to help administer the Sybase ASE database. These are some of the tools/commands:

- `isql`, a simple tool to access the server
- `sp_configure`, the system stored procedure to change the server configuration
- `sp_dboption`, the system stored procedure to change the database options
- `dbcc`, the database consistency checker commands to check the data consistency

- 'set' commands, to set a wide variety of server, database, table and session options like:
 - chained (use of begin transaction to start an implicit transaction)
 - transaction isolation level (to choose for example to do dirty reads)
 - showplan, statistics io, statistics time (to monitor query performance)

Other DBA tasks, that needs to be performed routinely:

Database Consistency Checks:

Running database consistency checks is a standard DBA procedure to determine if data may be invalid because of corruption at the database level. These are the different options to consider:

- dbcc checkdb: Checks integrity of data and index pages in a database.
- dbcc checkalloc: Database wide allocation check
- dbcc checkcatalog: Database system catalogs consistency check
- dbcc checkstorage: checks the consistency of database objects (combines checks above) without blocking access by other tasks

Consistency checkers may be resource intensive, consider, in case of large databases, setting up a schedule where system catalogs and user tables are check in succession.

Archiving data:

As the database grows, archiving some of the historical data in another database or on files may be considered. Queries on a very large table can potentially be much costlier than on small tables especially where there is no index available for the Sybase ASE optimizer to use. This can have application consequences and needs to be designed into the application.

Threshold checking:

Automatic last chance threshold on the devices can be programmed. Sybase devices may become full if not monitored regularly (see System Administration Guide, Chapter 15: Managing free space with thresholds). The last chance threshold can also be used to automatically dump the transaction log and prevent any process from being suspended because of the log being full.

Reorganizing table and index space:

Over time, random inserts and deletes can cause page fragmentation and inefficient space utilization. This will cause I/Os to be more random and key order scans to be slower. Sybase utilities are available to reorganize the data and index pages:

- All-page lockscheme, drop/recreate index (Drop requires exclusive table lock, Create requires shared table lock)
- Data-only locking scheme (datarows and datapages)
 - Rebuild table using reorg rebuild tablename (Requires exclusive table lock)
 - Compact index using reorg reclaim_space (Compacts pages but does not affect page chain clustering, no exclusive table lock)
 - Re-cluster index using reorg rebuild tablename indexname (Compacts pages and re-cluster page chain, No exclusive table lock)

Refreshing the table statistics:

For the Sybase ASE optimizer to choose the right query plans there need to be up-to-date statistics available. These are stored in the system table sysstatistics and systabstats. The command used to update the table statistics is “update statistics <objectname>”

Statistics can be updated at the table, index or column level. The frequency should be based on how quickly the data distribution changes.

Monitor space usage:

System stored procedures are available to manually display space usage for a database, a table or a segment (sp_spaceused, sp_helpsegment <segmentname>). Consider setting up user-defined thresholds on your data segments to alert you when space available is below a certain threshold.

There are several software tools that can help and facilitate basic administration tasks. These include Adaptive Server Monitor, used for monitoring server performance and other activities, and Sybase Central, which simplify many administration tasks. There are also many third-party software packages available designed to help System Administrators manage daily maintenance activities.

4 ADMINISTRATIVE DIFFERENCES

In this section, the administrative differences between the two products are considered.

4.1 Storage Management and Database Creation

MSSQL maps a database to a distinct set of physical data files and filegroups. ASE utilizes database devices (e.g., operating system files, Windows disk partitions, or Unix raw partitions) where each device can be used by one or more databases, and a database can use space from one or more devices. When creating a database the DBA chooses exactly how much space is to be used from each device. Logical areas of storage are managed via file groups within MSSQL and segments within ASE databases.

MSSQL can be configured to silently increase the size of its disk file (data and log) to accommodate growth using the FILEGROWTH clause of the CREATE DATABASE command. ASE cannot increase the size of its database devices automatically, but they can be manually resized larger at a later time. MSSQL also allows for a database to be resized lower whereas ASE does not.

Both products benefit from the performance and recoverability features of RAID devices. In addition, ASE has its own built-in mirroring commands for environments without RAID or Logical Volume Management (LVM).

4.2 Database Transaction Log

Fundamentally both products have similar transaction logging mechanisms, with each database having its own write-ahead transaction log to capture database events, such as inserted, updated and deleted rows as well as allocation and deallocation of data and index pages.

The MSSQL transaction log is implemented as a set of virtual logs on one or more physical data files which auto-grow in a round-robin fashion. The log files can also be set to shrink automatically to reclaim space after a backup.

Within ASE, the transaction log is an integrated part of the database and is actually a system table called syslogs created on its own segment named logsegment. The devices for the segment logsegment are specified at database creation, but may be changed later via ALTER DATABASE or sp_logdevice. The DBA can decide whether the transaction log, either for performance or recoverability, should be stored on separate database devices from the rest of the database's data.

4.3 Database Server Configuration

Both ASE and MSSQL use the `sp_configure` system procedure to specify server-wide options, but the set of configuration options differs between the two products. For a comparison of configuration parameters, see Server Configuration Option Differences in Appendix B.

MSSQL requires the RECONFIGURE command as a necessary last step to activate the changed parameter(s). Most ASE parameters are dynamic, meaning they take effect immediately after executing `sp_configure`. For both servers, a stop and restart may be required for certain static parameters to take effect.

Both products store current configuration parameters in the `sysconfigures` table, while current runtime values are reflected in `syscurconfigs`. In ASE, a text configuration file also stores the current configuration settings, conventionally called "`SERVER_NAME.cfg`" for an ASE server named "`SERVER_NAME`". This file is updated each time a parameter is changed with `sp_configure` and it can also be edited manually. The configuration file may be used to share configuration settings between servers, to start servers with different pre-set configurations, or to start a server with a previously known good configuration file should the server fail to properly start after a parameter change.

4.4 Setting Database Options

To set database options, ASE uses the `sp_dboption` procedure. MSSQL still supports this procedure but is phasing it out in favor of enhancements to the ALTER DATABASE command. The ASE version of this procedure requires a CHECKPOINT command to be run in the database in order for the changed options to take effect. For a comparison of database configuration options, see Database Option Differences in Appendix B.

4.5 Data Import and Export

Both products provide facilities for importing data from and exporting data to files. The `bcp` (Bulk Copy) utility provided by both products shares some common options including exporting the data using a native binary format (the `-n` option) or human-readable character format (the `-c` option).

MSSQL includes Data Transformation Services (DTS) to transfer data from tables or queries between OLE DB and ODBC data sources. These services are not available natively in ASE, however Sybase partners with companies to provide Extract, Transform, and Load (ETL) services.

Unfortunately the native binary files produced by the `bcp -n` native option cannot be reliably used to migrate data from MSSQL to ASE due to differences in how the two products store data values. Therefore the `-c` character option should be used to create portable and human-readable values for number, binary, and datetime datatype values. ASE provides direct data import/export using CIS and the services of Enterprise Connect / Data Access (ECDA).

4.6 Job Scheduling and Alerts

MSSQL works in conjunction with the SQL Server Agent Service to schedule jobs, send alerts, and interact with operators (people assigned to manage the server). Scheduling of scripts or T-SQL batches for onetime or repetitive execution is configured using Enterprise Manager, SQL-DMO (Distributed Management Objects), or T-SQL (e.g. `sp_add_job`). SQL Server Alerts is a facility offered by SQL Server Agent which constantly monitors the Windows application log for MSSQL events and can perform some action in response to their occurrence.

Job Scheduler of Adaptive Server, allows you to create and schedule jobs, and also share jobs and schedules. One database administrator can create a job and other database administrators can then schedule and run that job on another server. Jobs can be created from scratch, command line or GUI, loaded from a SQL batch file, or generated from a predefined template.

Job Scheduler captures the results and output of jobs, and records that information in log tables. Job Scheduler keeps a history of scheduled jobs. However, to keep a limit on the size of the history table, Job Scheduler is self-monitoring and removes outdated, unnecessary history records.

Job Scheduler is comprised of the following components:

- An internal Adaptive Server task
- An external process called the JS Agent
- The *sybmgmtdb* database and stored procedures
- The graphical user interface
- Predefined templates from which the database administrator may create and schedule useful, time-saving jobs

The internal ASE task determines when scheduled jobs should run and creates a historical record of jobs that are run. It starts the JS Agent process and feeds JS Agent the necessary information to retrieve job information and run the job on the specified ASE.

The JS Agent retrieves the job information from Job Scheduler's own database, called *sybmgmtdb*. Then it logs in to the target ASE and issues the job commands. When the job completes, JS Agent logs any result or output to the log tables in the *sybmgmtdb* database.

All the job, schedule, and scheduled job information, and data needed by the JS task for internal processing is stored in the *sybmgmtdb* database. Most access to data in the *sybmgmtdb* database is via stored procedures. The stored procedures make the data available to the GUI, the JS Agent and the commandline interface. Only the JS task accesses data directly from the *sybmgmtdb* database.

The GUI assists the user in creating and scheduling jobs, viewing job status and job history and in controlling jobs. The GUI also provides an administration feature to turn on and off the ASE internal task and therefore the ability of Job Scheduler to process and execute scheduled jobs.

Templates are an important tool in defining tasks for self-management of the database, such as database backups, reorganization rebuilds, modification of configuration parameters, and statistics updates and monitoring. They are implemented as batch Transact-SQL commands for which parameter values can be provided. Database administrators can use templates to generate jobs, which may then be scheduled to run at desired times.

4.7 Space Monitoring

MSSQL sends events to the Windows application log when data and logfiles grow or shrink. If an alert has been defined for that event, the SQL Server Agent can respond by notifying operators, executing a job, or forwarding the event to a remote Windows server. Files and filegroups are usually configured to autogrow as more space is required, so monitoring for free space often comes down to looking at the free space left on the Windows filesystem. One method often used to monitor free disk space is to write a custom job that the SQL Server Agent executes periodically.

ASE provides a built-in ability to monitor and respond to fluctuations in database space via its free space thresholds feature. For each database transaction log there is a last-chance threshold to monitor the minimum amount of space required for dumping the transaction log before it fills completely. A user can also define additional free space thresholds on any segment in the database. When thresholds are crossed, the server will automatically execute the designated stored procedure. These procedures can execute remote and extended stored procedures, invoke shell scripts, email the DBA, add more disk space, or dump the transaction log as appropriate.

4.8 Open Transaction Monitoring

MSSQL's DBCC OPENTRAN command identifies the oldest active transaction and the oldest distributed and non-distributed replicated transactions, if there are any, in each database. ASE stores the equivalent oldest active transaction information in the table master..syslogshold. In addition, all active ASE transactions are stored in the table master..systransactions.

4.9 Auditing

MSSQL provides a graphical tool for monitoring any kind of database event using the monitoring tool SQL Profiler, which sends a trace of user and server events to a file or table. MSSQL also has audit modes to analyze classes of events ("c2 audit mode" option).

ASE has a comprehensive and powerful C2-compliant audit subsystem that allows audit trails of server activity to be captured at whatever level is appropriate for the organization. The audit subsystem has its own set of system procedures (e.g. `sp_audit`) and the sybsecurity database to capture and manage the audit trail.

4.10 Server Trace Flags

Both products offer a number of trace flags that are used to alter the behavior of the server in certain ways. Many of the commonly used trace flags have virtually identical behavior between the two products (for example 302, 310, 3604 and 3605). Others (such as 1204), while not identical, cause similar information to be displayed. As trace flags are low-level internal switches that often change from one release to another, it should not be assumed that any particular trace flag would work identically across both products. Sybase Customer Service and Support can provide advice on the use of particular trace flags.

4.11 System Stored Procedures

The MSSQL product adds several of its own unique system procedures to the original set inherited from Sybase SQL Server. Examples include `sp_dbremove`, `sp_helpsql` and `sp_procoption`. 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 system procedures is supplied in DBA Command Differences.

4.12 Character Set Support

There are some slight differences in how each product handles character set and Unicode support. MSSQL allows character set and sort order combinations (collations) to be set at the database, table, and even the column level. Unicode is supported in addition to any other chosen collations.

ASE allows one character set and sort order to be set at the server level, but can also support many languages for system and user messages and default date format as required by multi-lingual applications. Unicode support is also provided through the UTF-8 character set, including support for UTF-16 encoding for `unichar` and `univarchar` datatypes. The `COMPARE()` and `SORTKEY()` functions enable application developers to work in sort orders other than the server's default.

4.13 `sp_tableoption` Procedure

MSSQL provides this stored procedure to allow tables to be marked as RAM resident or to be marked so that bulk loading will acquire a table lock rather than multiple rowlocks. With ASE you can define named data caches and then bind tables, indexes and databases to those caches for a similar effect. For bulk loading, ASE reduces locking problems by providing for automatic lock promotion from row or page locks to a single table lock.

4.14 Start-up Procedures

MSSQL allows certain stored procedures to be marked as start-up procedures (via `sp_procoption`), which the server will run automatically after it has started. ASE does not offer a direct equivalent but a DBA can augment the startup script on Unix or Windows to execute ASE commands or procedures.

4.15 The DBCC Command

Both products offer the `dbcc` command for consistency checking of databases and other system level operations. Both offer parallel DBCC capabilities, running on as many processors as possible with multiple DBCC commands running at once. For MSSQL, DBCC is no longer required for regular maintenance except as a check before major system changes. For ASE, DBCC is still recommended for regular maintenance and has been extended with the `dbccdb` system database to check and fix page linkage, pointers and page allocation errors using DBCC `checkstorage`.

Keep in mind that due to its system level nature, the DBCC subcommands differ slightly between the two products. While core sub-commands (such as checkalloc) are the same, both products offer their own subcommands; DBA scripts that use commands specific to one product (such as MSSQL's DBCC SHRINKDB) may need to be modified before they can be used with ASE. A list of potential problems is provided in DBA Command Differences in Appendix B.

APPENDIX A – DATA TYPE MAPPING

MSSQL	Sybase	Comments
Bigint	Bigint	
Int	Int	
Smallint	Smallint	
Tinyint	Tinyint	
Bit	Bit	
Decimal	Decimal	
Numeric	Numeric	
Money	Money	
Smallmoney	Smallmoney	
Float	Float	
Real	Real	
Datetime	Datetime	
Smalldatetime	Smalldatetime	
Char	Char	MSSQL allow lengths up to 8000 characters while ASE's character allows length up to 16384 bytes, depending on the logical page size used to create the database and the locking scheme for the table.
Varchar	Varchar	MSSQL allow lengths up to 8000 var. characters while ASE's var. character allows length up to 16384 bytes, depending on the logical page size used to create the database and the locking scheme for the table.
Text	Text	
nchar	Nchar	MSSQL allow lengths up to 4000 var. characters while ASE's character allows length up to 16384 bytes, depending on the logical page size used to create the database and the locking scheme for the table.
Nvarchar	Nvarchar	MSSQL allow lengths up to 4000 characters while ASE's var. character allows length up to 16384 bytes, depending on the logical page size used to create the database and the locking scheme for the table.
Ntext	Text	

MSSQL	Sybase	Comments
Binary	Binary	MSSQL allow lengths up to 8000 characters while ASE's character allows length up to 16384 bytes, depending on the logical page size used to create the database and the locking scheme for the table.
Image	Image	
Varbinary	Varbinary	MSSQL allow lengths up to 8000 characters while ASE's character allows length up to 16384 bytes, depending on the logical page size used to create the database and the locking scheme for the table.
timestamp	timestamp	
Table	No Direct Equivalent	To simulate the table datatype, the application will need to be reworked, perhaps using temporary tables to store intermediate results.

APPENDIX B – SQL LANGUAGE DIFFERENCES

In this appendix, several lists are presented describing the development feature differences in the T-SQL language that are likely to be encountered when migrating applications from Microsoft SQL Server to Sybase Adaptive Server Enterprise.

Behavior, Limits, and Additional Components

- **DDL in Transactions**

By default, MSSQL allows Data Definition Language (DDL) statements, such as CREATE TABLE, to be part of a transaction. This feature can lock system tables, such as sysobjects, resulting in performance degradation. By default, this feature is disabled within ASE and may be turned on for a database using the “allow ddl in tran” database option.

- **Altering Objects**

MSSQL allows a stored procedure, trigger or view to be updated in place via the ALTER PROCEDURE, ALTER TRIGGER, and ALTER VIEW statements. This is not possible in ASE; objects must be dropped and then re-created when they need to be changed.

The abilities inherent in the ALTER TABLE syntax (adding/dropping/changing columns, types, defaults) are fairly compatible between the two products. There are a few features not present in one or the other. MSSQL's ALTER TABLE...ALTER COLUMN COLLATE syntax allows you to specify a collation sequence for a column; this feature is not supported in ASE. ASE's syntax allows properties related to table partitioning and storage to be specified; these are not supported in MSSQL. For a more complete list of differences, see Appendix A and T-SQL Statement Differences.

- **Name Resolution in Stored Procedures**

MSSQL 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 requires all referenced objects exist when the procedure is created.

- **Bound Connections**

MSSQL provides for a bound connection where a transaction may be separated from a user connection allowing for multiple user connections to participate in (be bound to) the same database transaction. ASE provides some of this functionality through the ODBC driver interface and also through XA-compliant transaction servers using ASE's Distributed Transaction Manager facility (DTM). DTM provides the most compatibility as a migration alternative to bound connections for use of the “suspend” and “join” semantics of XA-compliant transaction managers.

- **Object Identifiers**

MSSQL allows up to 128 characters in an object identifier, such as a table, column, or view name. ASE currently enforces the SQL92 specification of a maximum of 30 characters. MSSQL allows identifiers that have non-standard characters or are reserved keywords to be enclosed using double quotation marks or square brackets (as in “SELECT * FROM “Table X” where [order] = 123”). ASE currently only supports double quotation marks using the SET quoted_identifier option. Sybase expects that a future release of ASE will support 128 character identifiers as well as bracketed identifiers.

Most MSSQL servers are case-insensitive, allowing objects to be referenced using upper, lower or mixed case (e.g., “AUTHORS” is the same as “Authors” or “authors”). ASE servers can be installed using either a case sensitive or case-insensitive default sort order. If the ASE server uses a case-sensitive sort order, such as “binary”, then object identifiers are also case-sensitive (e.g., “AUTHORS” is different from “Authors”). To ensure portability, use a case-insensitive sort order in the ASE server, such as “nocase”.

- **Table Limits**

MSSQL allows up to 256 tables to be included as joins or sub queries in a single SELECT statement. ASE allows up to 64 tables for a single query. Similarly, MSSQL allows 1024 columns in a table whereas ASE allows 254 for variable length allpages-locked tables and 1024 for datapages and datarows tables.

- **Object Storage Parameters**

MSSQL uses the “fillfactor” server configuration option to influence how full the index pages should be when the index is created. This parameter can also be specified during CREATE INDEX but MSSQL recommends only using the server-wide configuration for fillfactor as it believes this default is best for most cases. ASE offers FILLFACTOR as well as MAX_ROWS_PER_PAGE, EXP_ROW_SIZE, and RESERVEPAGEGAP options to the CREATE INDEX and CREATE TABLE commands.

- **Indexed Views**

MSSQL provides for an index to be created on a view. This feature creates a snapshot of the data that a view describes at the time of index creation, making the view no longer virtual. MSSQL must keep the data synchronized via the association made through the SCHEMABINDING clause of the CREATE VIEW command. ASE does not provide this functionality.

- **Computed Columns**

ASE does not currently provide the MSSQL computed columns feature of the CREATE TABLE command. A computed column is a virtual column that is defined through an expression involving one or more other columns in the same table, a function, or another column. For now, a computed column in ASE can be simulated using an actual column and a trigger or some other method. Sybase expects that a future release of ASE will support computed columns.

Language, Functions, Procedures and Commands

- **Identity and Uniqueness** – Both products offer the identity column attribute that indicates that the server should assign a unique value for this column whenever a row is inserted. The MSSQL implementation allows a seed and increment to be specified and the column's datatype may be either tinyint, smallint, int, decimal(p,o) or numeric(p,o). MSSQL generates a value to insert into a table by adding the increment to the last inserted value. In contrast, ASE does not allow a seed or increment to be specified (although the DBA can place a seed value in the table using set identity_insert) as the seed and increment are always 1. An identity column's datatype must be numeric(p,o). ASE caches a block of identity values in memory (instead of generating them from disk one at a time) in order to increase performance. Because this block of values is memoryresident, it can be lost when the database is suddenly or abnormally shut down, resulting in gaps between identity values. To prevent large gaps, you can specify the block size to cache using the WITH IDENTITY_GAP option of the CREATE TABLE and SELECT INTO commands. Identity values can be explicitly inserted or updated using the commands set identity_insert and set identity_update.

The MSSQL identity column can be referred to by the pseudo name IDENTITYCOL whereas the ASE version uses the pseudo name SYB_IDENTITY. ASE has the “auto identity” database option to automatically include the SYB_IDENTITY column in each table that is created without one.

MSSQL provides a uniqueidentifier datatype (to support the use of Microsoft GUID identifiers). ASE does not support this datatype. An application that makes use of identity columns may need to review its usage for these differences. If it cannot be migrated directly, alternatives, like a custom counter using an integer column and an insert trigger, can be used instead.

- **ANSI Join Syntax** – MSSQL supports the ANSI 92 SQL join operators of JOIN, CROSS JOIN, INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN as well as the traditional T-SQL join syntax using ‘=’ for inner joins and ‘*=’ and ‘=*’ for left and right outer joins. ASE supports all the above clauses except CROSS JOIN and FULL OUTER JOIN.
- **SELECT Lock Syntax** – MSSQL allows the locking strategy for a SELECT statement to be specified via the NOLOCK, UPDLOCK, TABLOCK, PAGLOCK, and TABLOCKX qualifiers. ASE does not support this syntax, but provides alternatives using the HOLDLOCK, NOHOLDLOCK, and AT ISOLATION keywords as well as the LOCK TABLE command to lock a table in exclusive or shared mode.
- **Rowsets** – The MSSQL SELECT statement allows the use of pseudo-database objects called rowsets resulting from OLE DB queries. Rowsets are not supported in ASE.
- **BULK INSERT command** – MSSQL provides this command to perform a T-SQL version of the bcp utility. ASE does not provide this command; however bcp can be executed by using the system-supplied extended stored procedure xp_cmdshell. In addition, if the bulk insert is to be performed across servers, ASE can use the SELECT INTO EXISTING TABLE command to insert the data using the bulk insert protocol.
- **Cursors** – Both MSSQL and ASE implement cursors as defined by the SQL-92 standard; however MSSQL extends their capabilities and syntax in a number of proprietary ways. Migration problems caused by MSSQL language enhancements include use of global or local cursors as parameters to stored procedures. ASE allows neither global nor local cursors as parameters to procedures.

MSSQL provides the global variable @@FETCH_STATUS to monitor the status of the most recent cursor fetch. ASE provides @@SQLSTATUS with different status values. See Global Variable Differences in Appendix A for the value differences.
- **Variable Assignment** – The SELECT command in both servers is used for variable assignment (e.g., SELECT @variable = 123). In addition to SELECT, MSSQL provides SET (e.g. SET @variable = 123). SELECT is currently recommended for portability, however Sybase expects to offer the SET command for variable assignment in a future release.
- **INSERT with DEFAULT option** – MSSQL allows the DEFAULT option in the INSERT command VALUES clause to indicate that a default value be used when inserting into a table. ASE currently does not provide this syntax, so the ISNULL() function can be used instead.
- **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 ignores trailing spaces in the search pattern (i.e. “%A% “ is the same as “%A%”) whereas the MSSQL LIKE operator will attempt a literal match for the space. In ASE to search for trailing spaces, replace each trailing space in the pattern with “[]” (i.e., “%A%[]”).

System Functions, Global Variables, Session Settings, and Keyword Lists

ASE and MSSQL provide many common functions (e.g. GETDATE()), global variables (e.g. @@VERSION), session-based settings (e.g. SET statistics io), and keywords. For more information on the differences, please see Appendix A.

- **Extended Stored Procedures (ESPs)**

For MSSQL, ESPs are part of the Open Data Services layer between the relational engine and the server netlibraries. For ASE, ESPs are provided through a separate XP Server. While ASE offers ESPs in a similar manner to MSSQL, the set of system-supplied ESPs is slightly different. In particular, the MSSQL `xp_grantlogin` (also `sp_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 MSSQL ESPs (e.g. `xp_msver` and `xp_sprintf`) are not available in ASE, but many are implemented (`xp_cmdshell`, `xp_deletemail`, `xp_enumgroups`, `xp_findnextmsg`, `xp_logevent`, `xp_readmail`, `xp_sendmail`). Please see DBA Command Differences in Appendix B.

- **Optimizer Hints and Abstract Query Plans**

Both products allow optimizer hints to be supplied as part of the SELECT, UPDATE, INSERT and DELETE commands (e.g. "SELECT * FROM t1 (<hint>)", though the set of hints is different. MSSQL hints are mainly concerned with the locking and join strategies used. The ASE hints include information about indexes, pre-fetch I/O buffer pool sizes, parallelism, and data cache replacement strategies (aspects of query processing not relevant to MSSQL).

One advantage of ASE over MSSQL is the ability to specify the exact query plan the optimizer should use through abstract query plans. Abstract query plans decrease the optimization time as the decisions on join order, indexing, scanning methods, and many other components are specified in the plan. See "Introduction to Abstract Plans" in the ASE Performance and Tuning Guide. When migrating queries between databases, all hints should be removed and the queries re-tuned in the appropriate environment as required. If problems arise in the query optimization, abstract plans may be used.

- **RAISERROR Command**

The MSSQL RAISERROR command uses a similar construct as the C language `printf()` function for specifying the error message so datatype conversions can be explicitly controlled. The ASE RAISERROR uses a simpler syntax with placeholders (e.g. RAISERROR 20001 "Error in %1!", @var) and automatically converts argument types during variable substitution.

MSSQL provides a severity argument, WITH SETERROR clause, and WITH NOWAIT clause, which are not provided by ASE. MSSQL provides the WITH LOG clause to indicate that a copy of the message should be placed in the server log. In ASE, the `with_log` option can be specified when the message is added to the server via `sp_addmessage` for the same result. `sp_addmessage` also provides for adding the message in multiple languages.

- **Distributed Transactions**

To start a transaction that involves several MSSQL servers, the DISTRIBUTED clause must be added to the BEGIN TRANSACTION command to instruct the server to use MS DTC for transaction coordination. This clause is not used or necessary in ASE, as it will implicitly coordinate a distributed transaction between several ASE servers.

T-SQL Statement Differences

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

MSSQL T-SQL Statement	Sybase Equivalent	Comments
CUBE ROLLUP	SELECT ...GROUP BY	Replace aggregate operators with a number of "SELECT ... GROUP BY" statements for the aggregates that they actually need.
SELECT ... TOP ... [WITH TIES]	SET rowcount n	Use the ASE session level setting to return a portion of a result set.
SET @local_variable = expression	SELECT @local_variable = expression	Setting variables created through DECLARE should be done with SELECT syntax instead of SET.
DECLARE cursor_name SCROLL FETCH { NEXT PRIOR FIRST LAST ABSOLUTE RELATIVE }	declare cursor_name [semi_sensitive insensitive] [scroll no scroll] cursor for select_statement [for {read only update [of column_name_list]]	
DEALLOCATE cursor_name	DEALLOCATE CURSOR cursor_name	Cursor deallocation syntax for ASE includes keyword CURSOR.
GLOBAL cursors	No equivalent	ASE cursors are local in scope.
Global Temporary Tables: CREATE TABLE ##global_table	No equivalent	ASE does not support global temporary tables. A temporary table may be created in tempdb to simulate behavior. The table can be dropped prior to session termination.
uniqueidentifier data type and the corresponding rowguidcol pseudo column	Identity column, a binary column or an integer counter	uniqueidentifier is not available in ASE.
RAISERROR ((error_number error_message) severity, state [, argument_list] [WITH LOG NOWAIT SETERROR]	RAISERROR error_number ['error_message' [, argument_list] [WITH errordata restricted_select_list]	Adapt the RAISERROR statement: remove WITH clause options which are not used in ASE environment, replace printf()-style error_message with ASE's implicit conversion of @vars corresponding to numbered parameters in the error_message string.
FORMATMESSAGE()	To access error message text, select information from the system tables.	
sp_addmessage msgnum > 50000	Sp_addmessage msgnum msgnum > 20000	msgnum User messages in ASE are stored in sysusermessages while MSSQL stores them in sysmessages. ASE message numbers can be > 20,000

MSSQL T-SQL Statement	Sybase Equivalent	Comments
Optimizer hints of: INDEX NOLOCK UPDLOCK TABLOCK PAGLOCK TABLOCKX XLOCK HASH MAXDOP	Remove or replace with ASE's hints	Optimizer hints for tables and queries should be removed to allow the ASE optimizer to choose the optimal plan.
PRINT 'l' + 'am'	PRINT 'l %!l', 'am' DECLARE @s SELECT @s = 'l'+ 'am' PRINT @s	PRINT does not support concatenation of strings; use placeholders or a variable instead.
DEFAULT keyword in the VALUES list of an INSERT statement or stored procedure INSERT . . VALUES (DEFAULT)	Instead, use the ISNULL () function or define a default on the column and then do not even specify the column. For stored procedures, you can set a default for an parameters like: @input_param int = 1	Replace with appropriate syntax.
Derived tables and EXECUTE not available in VALUES list of INSERT	Remove and replace with a temporary table.	You cannot use a derived table (sub query) or the EXECUTE statement in ASE's INSERT.
CAST ()	CONVERT ()	Converting expressions datatypes.
GETANSINULL ()	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.	IF the database option "allow nulls by default" is set to true, 1 is returned, otherwise 0 is returned.
VAR () VARP () STDEV () STDEVP ()	No equivalent	These aggregate functions not available in ASE.
CURSOR_STATUS () APP_NAME () ISDATE () ISNUMERIC () PERMISSIONS () PARSENAME () CURRENT_TIMESTAMP () GETUTCDATE ()	No equivalent	These system functions not available in ASE.

MSSQL T-SQL Statement	Sybase Equivalent	Comments
STATS_DATE ()	Query system statistics tables directly	System function not necessary.
DATABASEPROPERTY () FILEPROPERTY () COLUMNPROPERTY () DATABASEPROPERTYEX () FILE_ID () FILE_NAME () FILEGROUP_ID () FILEGROUP_NAME () FILEGROUPPROPERTY () fn_listextendedproperty () FULLTEXTCATALOGPROPERTY () FULLTEXTSERVICEPROPERTY () INDEXKEY_PROPERTY () INDEXPROPERTY () OBJECTPROPERTY () SERVERPROPERTY () SESSIONPROPERTY () SQL VARIANT PROPERTY () TYPEPROPERTY ()	Query system catalog instead.	Meta-data functions not available.
UNICODE ()	ASCII ()	
QUOTENAME ()	No equivalent	
REPLACE ()	STR_REPLACE ()	
NCHAR ()	TO_UNICHAR ()	
LIKE operator involves trailing space in search argument	LIKE "ABC[][]"	ASE LIKE ignores trailing space; use "[][]" to represent trailing spaces.
BEGIN DISTRIBUTED TRANSACTION	BEGIN TRANSACTION	CIS will automatically coordinate multi-server transactions.
BEGIN TRANSACTION...WITH MARK	BEGIN TRANSACTION	tran_name Recovery using named points of work is not supported in ASE. Remove the WITH MARK syntax and use ASE point-in-time recovery.
SET IMPLICIT_TRANSACTION ON	SET chained ON	
IS_MEMBER () IS_SRVROLEMEMBER ()	MUT_EXCL_ROLES () PROC_ROLES () ROLE_CONTAIN () ROLE_ID () ROLE_NAME () SHOW_ROLE ()	Some ASE role functions not available in MSSQL.
Bracketed identifiers: SELECT * FROM [TABLE]	SET quoted_identifier ON and delimit identifiers using double quotes ("). SELECT * FROM "TABLE"	ASE does not support the use of square brackets for delimiting identifiers.
Identifiers > 30 characters	Identifiers <= 30 characters	Observe identifier length limits.

MSSQL T-SQL Statement	Sybase Equivalent	Comments
JOINS with > 64 tables	JOINS <= 64 tables	Observe the limit to the number of tables in a join.
Tables with > 254 columns	Tables <= 254 columns (allpages locking) Tables <= 1024 columns (datapages and datarows locking)	Observe limits to the number of columns in a table based on the desired locking scheme.
FOREIGN KEY... ON UPDATE DELETE { CASCADE NO ACTION}	Use triggers to cascade changes to other tables.	Cascading referential integrity does not exist in ASE.
ALTER TABLE...WITH NOCHECK	No equivalent	NOCHECK is default ASE behavior.
ALTER TABLE table_name ADD DROP COLUMN column_name...	ALTER TABLE table_name ADD DROP column_name...	Remove the keyword COLUMN when using ALTER TABLE to add or drop a column.
ALTER TABLE table_name ADD COLUMN column_name... WITH VALUES	ALTER TABLE table_name ADD column_name... DEFAULT	To add values to a new column for existing rows, use DEFAULT keyword of ALTER TABLE.
ALTER TABLE table_name ALTER COLUMN column_name...	ALTER TABLE table_name MODIFY column_name...	To change a column's datatype, default or nullability in ASE, use MODIFY keyword of ALTER TABLE.
ALTER TABLE ADD DROP ROWGUIDCOL	No equivalent	Since ASE does not support the uniqueidentifier datatype, these references will have to be removed and substituted with a different identifying attribute.
CREATE TRIGGER...INSTEAD OF AFTER	sp_settriggerorder	By default, ASE triggers are AFTER triggers: the trigger fires after the firing event. No equivalent ASE allows one trigger for each insert, update, and delete on a table. To achieve the effect of multiple triggers, place each of the trigger bodies in stored procedures and call these stored procedures from a single table trigger.
CREATE PROCEDURE/TRIGGER/VIEW... WITH ENCRYPTION.	sp_hidetext	To conceal query batch for the database object in the syscomments table, use the sp_hidetext procedure.
CREATE PROCEDURE... FOR REPLICATION	Use ASE Replicator or Sybase Replication Server for replication	Replication in ASE is handled through ASE Replicator or Sybase Replication Server for replication. Consult the product manuals for specific commands and usage

MSSQL T-SQL Statement	Sybase Equivalent	Comments
CREATE TABLE ...col_name AS computed_col_expression	Add a column to the table and maintain through a trigger or the application.	Computed columns are not supported in ASE.
CREATE VIEW ... federated_table ... UNION ALL federated_table	No equivalent	Partitioned views as used by member tables in Federated Servers are not available.
CREATE VIEW.. WITH SCHEMABINDING	No equivalent	Indexed Views are not available in ASE.
IDENTITY(seed, increment) datatype	datatype IDENTITY	For ASE, remove the seed and increment when defining identity columns.
IDENT_CURRENT(table_name) IDENT_SEED() IDENT_INCR()	NEXT_IDENTITY(table_name)	For ASE, the last identity value inserted for any table is retrieved with @@IDENTITY. To find the next available identity value that will be inserted for a table, use NEXT_IDENTITY().
@@IDENTITY	@@IDENTITY	Last identity value assigned in the session for any table in all scopes.
SCOPE_IDENTITY()	No equivalent	Last identity for any table in current scope.
IDENTITYCOL	SYB_IDENTITY	To refer to an identity column in a table.

ANSI System Value Functions

MSSQL	ASE Equivalent	Comment
CURRENT_TIMESTAMP		The MSSQL 'CREATE ALTER TABLE' syntax allows five ANSI functions to return a system supplied value.
CURRENT_USER	USER_NAME()	
SESSION_USER		
SYSTEM_USER	SUSER_NAME()	
USER	USER	

MSSQL-Specific Functions and ASE Equivalents

MSSQL	ASE Equivalent	Comment
SOME ()	IN ()	Subquery function to compare a single value with a list of values.
CHECKSUM () CHECKSUM_AGG ()	No Equivalent	Function returns checksum over values in a row or expression.
COLLATE ()	SORT_KEY()	Retrieve an alternative collation value for sorting.
COLLATIONPROPERTY ()	No Equivalent	Obtains properties of a collation sequence.
COUNT_BIG ()	COUNT ()	Similar to COUNT(), but returns bigint data type.
GETUTCDATE ()	No Equivalent	Retrieves Greenwich Mean Time.
OPENQUERY ()	No Equivalent	Passes query to a linked server. Create and use proxy tables instead.
OPENROWSET ()	No Equivalent	Creates a rowset from a remote OLE DB data source. Create and use proxy tables instead.
ROWCOUNT_BIG ()	ROWCOUNT ()	Similar to ROWCOUNT(), but with bigint return type.
SUSER_SID ()	SUSER_ID ()	Returns the login identifier.
SUSER_SNAME ()	SUSER_NAME ()	Returns the return login name.
DB_NAME ()	DB_NAME ()	Returns the database name
DATEADD ()	DATE_ADD ()	Returns a new datetime value based on adding an interval to the specified date.
DATEDIFF ()	DATEDIFF ()	Returns the number of date and time boundaries crossed between two specified dates.
DATEPART ()	DATEPART ()	Returns an integer that represents the specified datepart of the specified date.
UPPER ()	UPPER ()	Returns a character expression with lowercase character data converted to uppercase.
LOWER ()	LOWER ()	Returns an integer that represents the specified datepart of the specified date.

Additional ASE functions not available in MSSQL

ASE Function	Comment
Char_length()	Returns the number of characters in an expression.
COMPARE()	Compares two character expressions using the specified collation and sorting rules.
Curunreservedpgs	Returns the number of free pages in the specified disk piece.
DATA_PGS()	The number of pages used by specified table or index.
Difference()	Returns the difference between two soundex values.
is_quiesced	Indicates whether a database is in quiesce database mode. Is_quiesced returns 1 if the database is quiesced and 0 if it is not.
LICENSE_ENABLED()	A positive or negative value indicating whether specified licensed components (e.g., ASE_SERVER, ASE_HA, ASE_DTM, etc.) are enabled on the server.
LOCKSCHEME()	The locking scheme (allpages, datapages, datarows) as a string.
PTN_DATA_PAGES()	Number of data pages used by a partition.
Reserved_pages	Reports the number of pages reserved to a table, index or a specific partition.
Show_role	Shows the login's currently active system-defined roles.
SORTKEY()	Provides values for sorting using different character collations.
SYB_SENDMSG()	Sends a text message via UDP to the specified IP address and port.
Tempdb_id	Reports the temporary database to which a given session is assigned.
To_unichar	Returns a <i>unichar</i> expression having the value of the integer expression
tran_dumptable_status	Returns a true/false indication of whether dump transaction is allowed.
UHIGHSURR()	Detects high and low location of surrogate pair in a Unicode value.
ULOWSURR()	
USCALAR()	Returns scalar value for first Unicode character in an expression.
Used_pages	Reports the number of pages used by a table, an index, or a specific partition.
Valid_name	Returns 0 if the specified string is not a valid identifier or a number other than 0 if the string is a valid identifier, and can be up to 255 bytes in length.
Valid_user	Returns 1 if the specified ID is a valid user or alias in at least one database on this Adaptive Server.

Session Option Setting Differences

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

MSSQL	ASE Equivalent	Comment
ANSI_NULL_DFLT_ON ANSI_NULL_DFLT_OFF	No equivalent	Not available as a session level setting. Use the ASE database option "allow nulls by default".
ANSI_DEFAULTS ANSI_PADDING	No equivalent	Trailing spaces on variable length columns are always truncated, while fixed-length values are always padded with spaces.
ANSI_NULLS	Ansinull	
CURSOR_CLOSE_ON_COMMIT	close on endtran	
DEADLOCK_PRIORITY	No equivalent	
FIPS_FLAGGER	Fipsflagger	
IMPLICIT_TRANSACTIONS	Chained	
LOCK_TIMEOUT	Lock wait	
NUMERIC_ROUNDABORT	arithabort numeric_truncation	
QUERY_GOVERNOR_COST_LIMITS	Impose resource limits on users instead of through a session.	Use administrative resource limits on users via Resource Governor feature.
REMOTE_PROC_TRANSACTION	transactional_rpc	
SHOWPLAN_TEXT	showplan	There is no equivalent to MSSQL's SHOWPLAN_ALL which produces a result set rather than a human readable plan.
XACT_ABORT	Not available	Use T-SQL transactional control statements to define transaction boundaries.
STATISTICS_PROFILE	Not available	Use the ASE <i>optdiag</i> utility to show statistics on objects and data.

Additional ASE-Specific T-SQL Session Options

ASE SET Option	Comment
ansi_permissions	Forces checking for ANSI-SQL permissions requirements for delete and update statements.
ansinull	Impacts on both aggregate and comparison behaviors.
arithabort	Determines how Adaptive Server behaves when an arithmetic error occurs.
arithabort numeric_truncation	Specifies the Adaptive Server behavior following a loss of scale by an exact numeric type during an implicit datatype conversion.
arithignore	Determines whether Adaptive Server displays a message after a divide-by-zero error or a loss of precision.
chained	Begins a transaction just before the first data retrieval or data modification statement at the beginning of a session and after a transaction ends.
close on endtran	Causes Adaptive Server to close all cursors opened within a transaction at the end of that transaction.
fipsflagger	Determines whether Adaptive Server displays a warning message when Transact-SQL extensions to entry-level ANSI SQL are used.
quoted_identifier	Determines whether Adaptive Server recognizes delimited identifiers within double quotation marks
string_rtruncation	Determines whether Adaptive Server raises a SQLSTATE exception when an insert or update command truncates a <i>char</i> , <i>unichar</i> , <i>varchar</i> or <i>univarchar</i> string
transaction isolation level	Sets the transaction isolation level for the session. After setting this option, any current or future transactions operate at that isolation level.

Global Variable Differences

MSSQL Global Variable		ASE Equivalent		Comment
@@fetch_status		@@fetch_status		
Status	Description	Status	Description	
0	Successful FETCH.	0	Successful FETCH.	
-1	Cursor reached end of dataset or row does not exist.	-1	fetch operation unsuccessful.	
-2	Row deleted or key updated after cursor opened (scrollable option).	-2	value reserved for future use	Cursor FETCH status values are slightly different between servers.
@@MAX_PRECISION		@@MAX_PRECISION		Note: ASE's max precision is 38 digits.
@@REMSERVER		No equivalent		
@@SERVICENAME		No equivalent		
@@PROCID		@@PROCID		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 MSSQL version of @@PROCID will be set to the ID of the trigger in this case.

Reserved Keyword Differences

Reserved keywords in the T-SQL language should not be used as object identifiers (e.g. table or column names). Any database objects created with names matching any of these keywords will need to be renamed during migration (for example using an application prefix). Optionally, ASE provides a session option, `quoted_identifier`, which may be turned on to allow access to object names with reserved words.

ADD	DESC	KEY	ROLLBACK
ALL	DISK	KILL	ROWCOUNT
ALTER	DISTINCT	LIKE	RULE
AND	DROP	LINENO	SAVE
ANY	DUMMY	LOAD	SCHEMA
AS	DUMP	NATIONAL	SELECT
ASC	ELSE	NONCLUSTERED	SET
AUTHORIZATION	END	NOT	SETUSER
BEGIN	ERRLVL	NULL	SHUTDOWN
BETWEEN	ESCAPE	NULLIF	SOME
BREAK	EXCEPT	OF	STATISTICS
BROWSE	EXEC	OFF	TABLE
BULK	EXECUTE	OFFSETS	TEXTSIZE
BY	EXISTS	ON	TO
CASCADE	EXIT	OPEN	TRAN
CASE	EXTERNAL	OPTION	TRANSACTION
CHECK	FETCH	OR	TRIGGER
CHECKPOINT	FILLFACTOR	ORDER	TRUNCATE
CLOSE	FOR	OUTER	TSEQUAL
CLUSTERED	FOREIGN	OVER	UNION
COALESCE	FROM	PLAN	UNIQUE
COMMIT	GOTO	PRIMARY	UPDATE
COMPUTE	GRANT	PRINT	USE
CONSTRAINT	GROUP	PROC	USER
CONTINUE	HAVING	PROCEDURE	VALUES
CONVERT	HOLDLOCK	PUBLIC	VARYING
CREATE	IDENTITY	RAISERROR	VIEW
CURRENT	IF	READ	WAITFOR
CURSOR	IN	READTEXT	WHEN
DATABASE	INDEX	RECONFIGURE	WHERE
DBCC	INSERT	REFERENCES	WHILE
DEALLOCATE	INTERSECT	REPLICATION	WITH
DECLARE	INTO	RETURN	WRITETEXT
DEFAULT	IS	REVERT	
DELETE	JOIN	REVOKE	

MSSQL's T-SQL language reserved keyword list includes the following keywords that are not in ASE's list.

COLLATE	FILE	OPENQUERY	SYSTEM_USER
COLUMN	FREETEXT	OPENROWSET	TABLESAMPLE
CONTAINS	FREETEXTTABLE	OPENXML	THEN
CONTAINSTABLE	FULL	PERCENT	TOP
CROSS	FUNCTION	PIVOT	UNPIVOT
CURRENT_DATE	IDENTITY_INSERT	PRECISION	UPDATETEXT
CURRENT_TIME	IDENTITYCOL	RESTORE	BACKUP
CURRENT_TIMESTAMP	INNER	RESTRICT	DOUBLE
CURRENT_USER	LEFT	RIGHT	
DENY	NOCHECK	ROWGUIDCOL	
DISTRIBUTED	OPENDATASOURCE	SESSION_USER	

APPENDIX C – SYSTEM TABLE COMPARISON

Many system tables from earlier releases are implemented as a set of views in SQL Server 2005 for. This section displays the mapping of system views of MSSQL to the system tables of ASE. Though they share many of the same tables/views in master and other databases, the details of each table are slightly different. For more information on each, see SQL Server Books Online for MSSQL and the Reference Manual for ASE.

Master Database:

The following table maps the system tables that are in the **master** database of ASE to their corresponding system view in SQL Server 2005.

System View of MSSQL	System Table of ASE	Description
sys.master_files	Sysdevices	
sys.dm_exec_cached_plans sys.dm_exec_plan_attributes sys.dm_exec_sql_text	N/A	No direct equivalent in ASE
sys.syscharsets	Syscharsets	Character Sets or Sort orders
sys.configurations	Sysconfigures	Configuration Parameter in effect
sys.databases	sysdatabases	Database within the server
sys.backup_devices	sysdevices	Database disk devices (files or row partitions) and tape and disk dump devices for ASE. Database file information for MSSQL
N/A	sysengines	Information about the ASE engine process
sys.syslanguages	Syslanguages	Language known to server
N/A	Syslisteners	Active network connection Listeners
sys.dm_tran_locks	Syslocks	Current active locks
N/A	Sysloginroles	Roles assigned to logins

System View of MSSQL	System Table of ASE	Description
sys.server_principals sys.sql_logins (Transact-SQL)	Syslogins	Login, Password and Authentication Information
N/A	Syslogshold	Oldest open transaction and Replication transaction point
sys.messages	Sysmessages	System error or warning messages
sys.dm_os_performance_counters	Sysmonitors	Performance Information
sys.linked_logins	–	Information about user mapping with linked server
sys.dm_exec_connection sys.dm_exec_sessions	Sysprocesses	Connected user and background .system processes
sys.remote_logins	Sysremotelogins	Remote logins and passwords
N/A	Sysresourcelimits	Resource Limits
N/A	Syssecmechs	Security services for each security mechanism
N/A	Sysservers	Local and remote servers
sys.servers	Syssessions	User connections for ASE High Availability option
N/A	Syssrvrole	Server-wide system and user-defined roles
N/A	Systimeranges	Defined time ranges for the resource limits
N/A	Systransactions	Active Transaction
N/A	Sysusages	Disk pieces (portions of a device allocated to a database)

All Database:

System View of MSSQL	System Table of ASE	Description
N/A	Sysalternates	ASE login alias to existing database users
N/A	Sysattributes	Object attributes
sys.columns	Syscolumns	Columns in tables and views, and parameters in procedures.
sys.sql_modules	Syscomments	SQL text for views, rules, defaults, triggers, and procedures
sys.check_constraints sys.default_constraints sys.key_constraints sys.foreign_keys	Sysconstraints	Referential and check constraints for tables and columns
sys.sql_dependencies	Sysdepends	Procedure, view, or table referenced by procedure, view, or trigger
sys.filegroups	N/A	Filegroups in database. ASE equivalent is in <code>syssegments</code> , <code>master..sysdevices</code> and <code>master..sysusages</code>
sys.database_files	N/A	Files in a database. ASE equivalent is in <code>master..sysdevices</code> and <code>master..sysusages</code>
sys.fulltext_catalogs	N/A	Full-text catalogs for database
N/A	Sysgams	
sys.indexes sys.partitions sys.allocation_units sys.dm_db_partition_stats	Sysindexes	Clustered or nonclustered indexes, tables without clustered indexes, tables containing text or image data
sys.index_columns	N/A	Columns in index; ASE equivalent is stored in <code>sysindexes</code>
N/A	Sysjars	Java archive (JAR) files
N/A	Syskeys	Primary, foreign, or common key set by user
N/A	Syslogs	Transaction log
sys.database_role_members	N/A	Members of database roles
sys.objects	Sysobjects	Tables, views, procedures, rules, triggers, defaults, and (in tempdb only) temporary objects
N/A	Syspartitions	Partition (page chain) of a partitioned table
sys.database_permissions sys.server_permissions	N/A	Permissions granted to users, groups, and roles. ASE equivalent table is <code>sysprotects</code>
N/A	Sysprocedures	Query tree (binary compilation) for views, rules, defaults, triggers, and procedures

System View of MSSQL	System Table of ASE	Description
sys.database_permissions sys.server_permissions	Sysprotects	User permissions information through GRANT and REVOKE
N/A	Sysqueryplans	Abstract query plans and SQL text
sys.foreign_keys	Sysreferences	Referential integrity constraints declared on tables and columns
N/A	Sysroles	Server role to local database group associations
N/A	Syssegments	Segments (named collection of disk pieces)
N/A	Sysstatistics	Statistics for columns
N/A	Systabstats	Statistics for tables and indexes
N/A	Systhesholds	Thresholds defined for the database's segments
sys.types	Systypes	System-supplied and user-defined datatypes
N/A	Sysusermessages	User-defined messages
sys.database_principals	Sysusers	Users allowed in the database
N/A	Sysxtypes	Extended SQL and Java class-based datatypes

