TECHNICAL WHITE PAPER

MySQL to Sybase® ASE Migration Guide

SYBASE, INC.

FEBRUARY 2007

TABLE OF CONTENTS

1 INTRODUCTION

MySQL is a open source relational database management system, is developed, distributed, and supported by MySQL AB. Whereas Sybase ASE (Adaptive Server Enterprise) is a relational database management system manufactured and sold by Sybase Inc. For various reasons users of MySQL may need to migrate their applications to Sybase 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.

By migration, in this context, we mean the process of changing an application so that it uses ASE, rather than MySQL, as its underlying database management system; this involves moving the schema and data from MySQL 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. This document is primarily intended to assist with the application migration process from MySQL to Sybase ASE. This document discusses in detail the differences between Adaptive Server Enterprise and MySQL, including architecture, data type differences, feature differences, and syntax differences. The white paper presents the overview of the differences between the two products, from the point of view of both the application developer and DBA, and then describes a straightforward and systematic process for performing a migration from MySQL to Sybase ASE.

The migration process between the two is relatively easy and involves the following steps:

- Creating a suitable ASE server.
- Migrating the application database architecture.
- Migrating the applications data.
- Ensuring the administrative and security procedures are appropriate for the new environment.
- Checking the application SQL will work in the Sybase ASE environment.
- Ensuring the 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 MySQL and ASE.

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

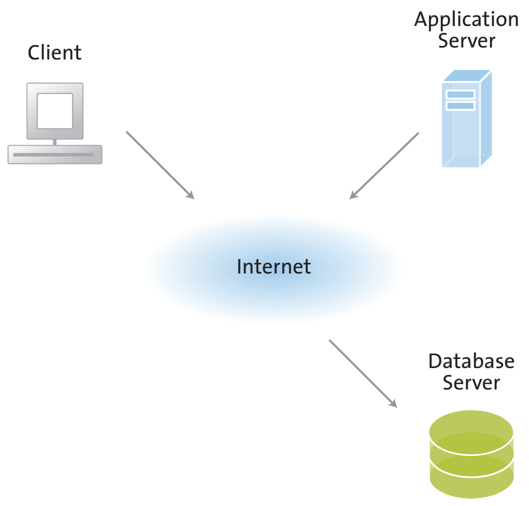
Some of the features for MySQL and ASE mentioned in this document are bundled with the respective software products and some features are purchased separately as add-on options. Please consult with your Sybase sales representative for the list of bundled features and add-on options available for each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

2 MIGRATION PROCESS

When migrating a MySQL application all the application components have to be checked for specific Sybase issues. The application may contain MySQL 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 the application to be migrated may look like:



The client will communicate through the Internet with the Application Sever® who will pass the messages to the Database server. All the components used have to be compatible with the Sybase Adaptive Sever Enterprise.

2.2 Migrating the data

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

- Creating the Sybase environment.
- Creating the administrative and security components.
- Moving the data.

The recommended tools for the migration are:

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

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 MySQL specific SQL.

There are five type of client application code that may need to be migrated. They consists 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, tuning and performance considerations.

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.
- Administrative procedures.
- Disaster recovery.
- Application performance obtained.

2.5 Optimize the Performance

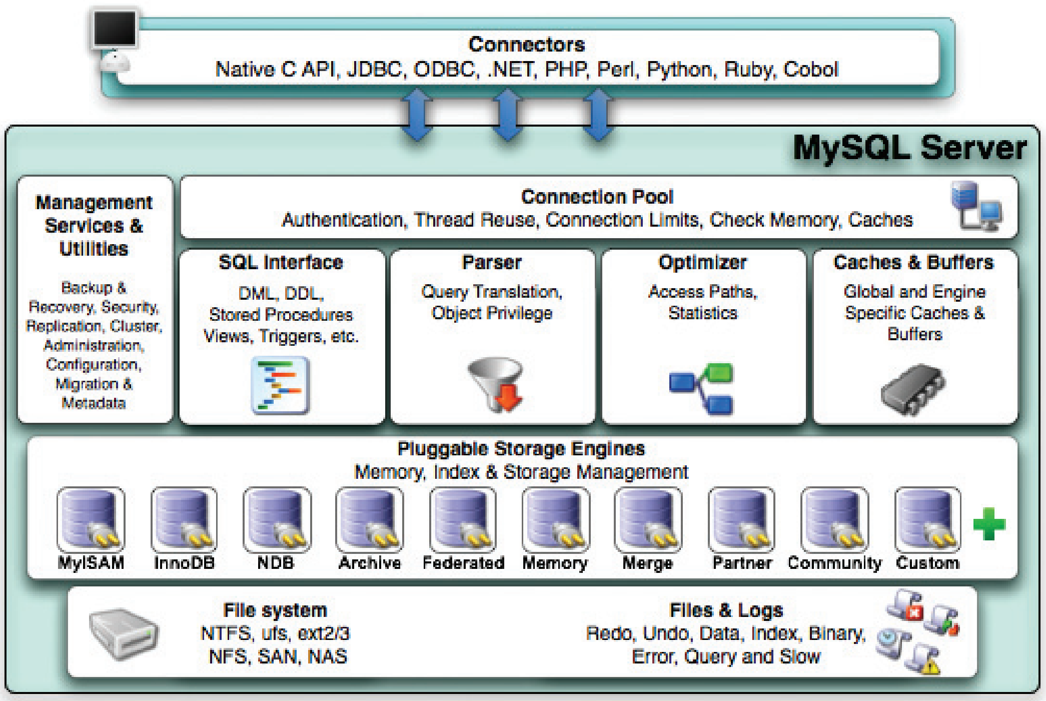
After the data and the application are migrated, check for the overall performance of the application. Derive performance of the application that is equivalent to or better than performance obtained by the SQL database.

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

3 ARCHITECTURE AND ADMINISTRATION

3.1 Architecture

MySQL


Connectors
Native C API, JDBC, ODBC, .NET, PHP, Perl, Python, Ruby, Cobol
MySQL Server
Management Services & Utilities
Backup & Recovery, Security, Replication, Cluster, Administration, Configuration, Migration & Metadata
Connection Pool
Authentication, Thread Reuse, Connection Limits, Check Memory, Caches
SQL Interface
DML, DDL, Stored Procedures Views, Triggers, etc.
Parser
Query Translation, Object Privilege
Optimizer
Access Paths, Statistics
Caches & Buffers
Global and Engine Specific Caches & Buffers
Pluggable Storage Engines
Memory, Index & Storage Management
MyISAM
InnoDB
NDB
Archive
Federated
Memory
Merge
Partner
Community
Custom
File system
NTFS, ufs, ext2/3
NFS, SAN, NAS
Files & Logs
Redo, Undo, Data, Index, Binary, Error, Query and Slow

MySQL is based on a tiered architecture, consisting of both primary subsystems and support components that interact with each other to read, parse, and execute queries, and to cache and return query results. The MySQL architecture consists of five primary subsystems that work together to respond to a request made to the MySQL database server:

- The Query Engine.
- The Storage Manager.
- The Buffer Manager.
- The Transaction Manager.
- The Recovery Manager.

The Query Engine

This subsystem consists of three interrelated components:

- The Syntax Parser
- The Query Optimizer
- The Execution Component

The Syntax Parser decomposes the SQL commands that it receives from the calling programs into a form that can be understood by the MySQL engine. That objects that will be used are identified, along with the correctness of the syntax. The Syntax Parser also checks the privilege levels for the objects that are being referenced.

The Query Optimizer checks to see which index should be used to retrieve data as quickly and efficiently as possible. It chooses one from among the several ways it has found to execute the query and then creates a plan of execution that can be understood by the Execution Component.

The Execution Component then interprets the execution plan and, based on the information it has received, makes requests of the other components to retrieve the records.

The Storage Manager

The Storage Manager interfaces with the operating system to write data to the disk efficiently. It writes to disk all of the data in user tables, indexes, and logs as well as the internal system data.

The Query Cache

The MySQL engine uses an extremely efficient result set caching mechanism, known as the Query Cache, that dramatically enhances response times for queries that are called upon to retrieve the exact same data as a previous query.

The Buffer Manager

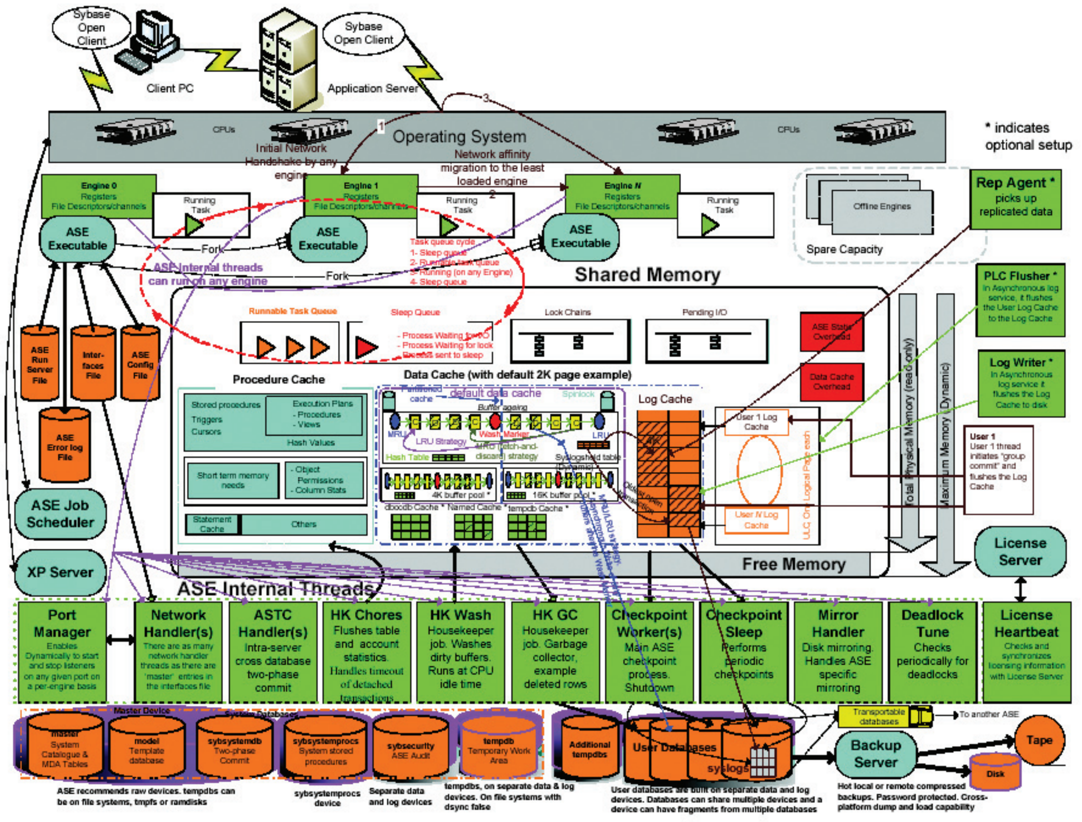
This subsystem handles all memory management issues between requests for data by the Query Engine and the Storage Manager.

The Transaction Manager

The function of the Transaction Manager is to facilitate concurrency in data access. This subsystem provides a locking facility to ensure that multiple simultaneous users access the data in a consistent way, without corrupting or damaging the data in any way.

The Recovery Manager

The Recovery Manager's job is to keep copies of data for retrieval later, in case of a loss of data. It also logs commands that modify the data and other significant events inside the database.


Sybase Open Client
Client PC
Application Server
Sybase Open Client
Operating System
CPUs
* indicates optional setup
Initial Network Handshake by any engine
Network affinity migration to the least loaded engine
Engine 0 Registers File Descriptors/channels
Engine 1 Registers File Descriptors/channels
Engine N Registers File Descriptors/channels
Running Task
ASE Executable
Fork
ASE Internal threads can run on any engine
Task queue cycle
1- Sleep queue
2- Runnable task queue
3- Running (on any Engine)
4- Sleep queue
Offline Engines
Spare Capacity
Rep Agent * picks up replicated data
Shared Memory
PLC Flusher * In Asynchronous log service, it flushes the User Log Cache to the Log Cache
Log Writer * In Asynchronous log service it flushes the Log Cache to disk
ASE Run Server File
Interfaces File
ASE Config File
ASE Error log File
Runnable Task Queue
Sleep Queue
- Process Waiting for I/O
- Process Waiting for lock
- Process sent to sleep
Lock Chains
Pending I/O
ASE State Overhead
Data Cache Overhead
Procedure Cache
Stored procedures
Triggers
Cursors
Execution Plans
- Procedures
- Views
Hash Values
Short term memory needs
- Object Permissions
- Column Stats
Statement Cache
Others
Data Cache (with default 2K page example)
default data cache
Buffer ageing
MRU
LRU
LRU Strategy
Wash Marker
MRU fetch-and-discard strategy
Hash Table
Syslogshold table
4K buffer pool *
16K buffer pool *
dbccdb Cache *
Named Cache *
tempdb Cache *
Log Cache
User 1 Log Cache
User N Log Cache
ULC On Logical Page each
Total Physical Memory (read only)
Maximum Memory (Dynamic)
User 1
User 1 thread initiates "group commit" and flushes the Log Cache
ASE Job Scheduler
XP Server
Free Memory
License Server
ASE Internal Threads
Port Manager Enables Dynamically to start and stop listeners on any given port on a per-engine basis
Network Handler(s) There are as many network handler threads as there are 'master' entries in the interfaces file
ASTC Handler(s) Intra-server cross database two-phase commit
HK Chores Flushes table and account statistics. Handles timeout of detached transactions
HK Wash Housekeeper job. Washes dirty buffers. Runs at CPU idle time
HK GC Housekeeper job. Garbage collector, example deleted rows
Checkpoint Worker(s) Main ASE checkpoint process. Shutdown
Checkpoint Sleep Performs periodic checkpoints
Mirror Handler Disk mirroring. Handles ASE specific mirroring
Deadlock Tune Checks periodically for deadlocks
License Heartbeat Checks and synchronizes licensing information with License Server
Master Device
System Databases
master System Catalogue & MDA Tables
model Template database
sybsystemdb Two-phase Commit
sybsystemprocs System stored procedures
sybsecurity ASE Audit
tempdb Temporary Work Area
Additional tempdbs
User Databases
syslogs
Transportable databases
To another ASE
Backup Server
Tape
Disk
ASE recommends raw devices. tempdbs can be on file systems, tmpfs or ramdisks
sybsystemprocs device
Separate data and log devices
tempdbs, on separate data & log devices. On file systems with dsync false
User databases are built on separate data and log devices. Databases can share multiple devices and a device can have fragments from multiple databases
Hot local or remote compressed backups. Password protected. Cross-platform dump and load capability

Adaptive Server runs as an application on top of an operating system and depends solely on the services exported by the operating system to function. Adaptive Server uses operating system services for process creation and manipulation, device and file processing, and inter-process communication. The hardware that the operating system runs on is completely transparent to Adaptive Server, which sees only the operating system's user interfaces.

Adaptive Server has virtual server architecture and has one or more operating system processes (engines) that service client requests. You can configure multiple processes to enhance performance on multiprocessor systems. A process is divided into two components, a DBMS component and a kernel component. The kernel component manages multiple tasks, while the DBMS component performs the database functions on behalf of the current task. Each client is associated with a task, and there are several tasks that perform specific services; for example, writing buffers or audit data to disk, and communicating with the network. The DBMS component of Adaptive Server manages the processing of SQL statements, accesses data in a database, and manages different types of server resources.

All server processes share two distinct areas of shared memory: a kernel area and a DBMS area. The kernel area contains data structures that relate to task management and operating system services, while the DBMS component contains caches of database buffers, object descriptors, and other resources used by the DBMS component of the server process.

The kernel layer of the server abstracts the operating system-specific functions so that the rest of the server has a consistent view of those functions regardless of the underlying operating system. Essentially, the kernel provides low-level functions, such as task and engine management, network and disk I/O, and low-level memory management to the rest of the server. The part of the server that processes a TDS request (the TDS engine) is part of the kernel layer.

The heart of Adaptive Server is the SQL/DBMS engine. The SQL/DBMS engine processes data manipulation statements (DML), utility commands, which are generally data definition statements (DDL) and administrative functions, and performs stored procedure executions. The DBMS engine uses the services provided by the resource manager routines and the access methods routines.

A resource manager is a set of subroutines that manage a particular system resource, such as pages, buffers, or a particular data structure. The DBMS engine uses these subroutines to manipulate various system resources. There are also a number of tasks within the server that perform specialized tasks. The resource managers, as well as these specialized tasks, rely on the kernel layer for low-level functions.

Access methods have two primary functions: the management of data on disk and manipulation of database constructs, such as tables and rows. Actual disk I/O routines are provided to the access methods routines by the kernel disk I/O routines.

Logical Database Design

Adaptive Server version 12.5 and later supports 2K, 4K, 8K, and 16K logical page sizes. Master devices built using Adaptive Server version 12.5 and later store information about its logical page size in the config block. The server's error log contains the system database's logical page size:

```
00:00000:00001:2001/06/07 19:27:44.80 server the logical pagesize of the server
is 2 Kb.
```

If you are rebuilding a corrupt system database in a server that was upgraded to version 12.5 and later, **dataserver** uses the default logical page size of 2K.

Physical Database Design

The ASE server does I/O to the raw devices or files; these are represented internally as virtual devices. A database can reside on one or spread out on many of these virtual devices, and a virtual device can hold many databases if you want. You should locate the OS-level device files on fast disks and make sure they are not removed or messed with by other applications or sysadmins on a cleanup crusade. The path to the virtual devices are stored in the master.sysdevices table, you can list these with the **sp_helpdevice**-stored procedure.

3.2 Structured Query Language

The Structured Query Language (SQL) is an open standard that has been maintained by the American National Standards Institute (ANSI). The implementation of this standard varies from vendor to vendor. SQL statements can be broken down into Data Definition Language (DDL) and Data Manipulation Language (DML). MySQL follows the ANSI standard but it has some variations. Some versions of MySQL do not support triggers, sub-queries, views, foreign key constraints etc.

Sybase ASE uses Transact-SQL (T SQL) as their data definition, data manipulation and data control language. It uses ANSI SQL-92 standard.

3.3 Memory Management

MySQL

MySQL perform database operations within memory as often as possible, rather than initiate more expensive disk access. The following section looks at some of the available variables that you can stipulate to determine your server's memory behavior:

Locking mysqld in Memory

As an administrator, you have the option to force the mysqld process to remain in the memory, rather than face the possibility of swapping to disk should system resources become scarce.

To lock *mysqld* in memory, launch it with the *–mmlock* option enabled. However, two conditions need to be met or this won't happen:

1. You must launch the *mysqld* process as a root.
2. Your operating system must support the ability of processes to perform this kind of operation.

Thread Memory Settings

Threads are the internal process mechanisms that MySQL uses to perform work as well as communicate with connected clients.

First, to determine your current thread status, use SHOW STATUS

```
mysql> SHOW STATUS LIKE '%THREAD%';

+-----------------------+-------+
| Variable_name          |Value|
+-----------------------+-------+
| Delayed_insert_threads |4|
| Slow_launch_threads    |0|
| Threads_cached            |9|
| Threads_connected      |27|
| Threads_created           |30|
| Threads_running           |18|
+-----------------------+-------+
```

To see what all your threads are doing, run SHOW PROCESSLIST or view MySQL's Administrator Thread tab.

Memory Tables

In MySQL storage engines, MEMORY (previously known as HEAP) tables are very fast, in-memory storage available for developers who need these capabilities. These tables are also used by MySQL itself for many smaller temporary tables.

As an administrator, you can decide how much memory you want to allocate for developer-generated tables by setting the *max_heap_table_size* variable. This value, measured in bytes tells MySQL how large one of these tables can become.

For server-generated temporary tables, use the *tmp_table_size* setting instead. It also tells MySQL how many bytes a temporary table can consume in memory before being swapped out to a disk based, MyISAM temporary table. These disk-based tables are obviously much slower than memory-based tables.

Binary Log and caching

Binary log serves many purposes, including essential support for transactions and replication. MySQL sets up a binary log cache for every connected client. As an administrator, you can configure the size of this buffer by setting the *binlog _cache_size* variable, which is applicable at the GLOBAL level.

You can also configure a not to exceed value for your binary log cache by specifying *max_binlog_cache_size*. However, the default of 4GB is probably sufficient for most of the environments.

Sybase ASE

How Adaptive Server allocates memory

All database object pages are sized in terms of the **logical page size**, which you specify when you build a new master device. All databases—and all objects in every database—use the same logical page size. The size of Adaptive Server's logical pages (2, 4, 8, or 16K) determines the server's space allocation. Each allocation page, object allocation map (OAM) page, data page, index page, text page, and so on is built on a logical page. For example, if the logical page size of Adaptive Server is 8K, each of these page types is 8K in size. All of these pages consume the entire size specified by the size of the logical page. Larger logical pages allow you to create larger rows, which can improve your performance because Adaptive Server accesses more data each time it reads a page. For example, a 16K page can hold 8 times the amount of data as a 2K page, an 8K page holds 4 times as much data as a 2K page, and so on, for all the sizes for logical pages.

The logical page size is a server-wide setting; you cannot have databases with varying size logical pages within the same server. All tables are appropriately sized so that the row size is no greater than the current page size of the server. That is, rows cannot span multiple pages.

Regardless of the logical page size for which it is configured, Adaptive Server allocates space for objects (tables, indexes, text page chains) in extents, each of which is eight logical pages. That is, if a server is configured for 2K logical pages, it allocates one extent, 16K, for each of these objects; if a server is configured for 16K logical pages, it allocates one extent, 128K, for each of these objects.

This is also true for system tables. If your server has many small tables, space consumption can be quite large if the server uses larger logical pages. For example, for a server configured for 2K logical pages, *systypes*—with approximately 31 short rows, a clustered and a non-clustered index—reserves 3 extents, or 48K of memory. If you migrate the server to use 8K pages, the space reserved for *systypes* is still 3 extents, 192K of memory. For a server configured for 16K, *systypes* requires 384K of disk space. For small tables, the space unused in the last extent can become significant on servers using larger logical page sizes.

Databases are also affected by larger page sizes. Each database includes the system catalogs and their indexes. If you migrate from a smaller to larger logical page size, you must account for the amount of disk space each database requires. Table below lists the minimum size for a database on each of the logical page sizes.

Table 3-1: Minimum database sizes

Logical page size	Minimum database size
2K	2MB
4K	4MB
8K	8MB
16K	16MB

The logical page size is not the same as the memory allocation page size. Memory allocation page size is always 2K, regardless of logical page size, which can be 2, 4, 8, or 16K. Most memory-related configuration parameters use units of 2K for their memory page size. These configuration parameters include:

- max memory
- total logical memory
- total physical memory
- procedure cache size
- size of process object heap
- size of shared class heap
- size of global fixed heap

Disk space allocation

The logical page size is not the same as the memory allocation page size. This is the unit in which disk space is allocated, and Adaptive Server allocates this space in 2K pages. Some of the configuration parameters use this 2K-page size for their allocation units.

Larger logical page sizes and buffers

Adaptive Server allocates buffer pools in units of logical pages. For example, on a server using 2K logical pages, 8MB are allocated to the default data cache. This constitutes approximately 2048 buffers. If you allocated the same 8MB for the default data cache on a server using a 16K logical page size, the default data cache is approximately 256 buffers. On a busy system, this small number of buffers might result in a buffer always being in the wash region, causing a slowdown for tasks requesting clean buffers. In general, to obtain the same buffer management characteristics on larger page sizes as with 2K logical page sizes, you should scale the size of the caches to the larger page size. So, if you increase your logical page size by four times, your cache and pool sizes should be about four times larger as well.

Adaptive Server typically allocates memory dynamically and allocates memory for row processing, as it needs it, allocating the maximum size for these buffers, even if large buffers are unnecessary. These memory management requests may cause Adaptive Server to have a marginal loss in performance when handling wide-character data.

Heap memory

A heap memory pool is an internal memory pool created at start-up that tasks use to dynamically allocate memory as needed. This memory pool is used by tasks that require a lot of memory from the stack, such as tasks that use wide columns. For example, if you make a wide column or row change, the temporary buffer this task uses can be as large as 16K, which is too big to allocate from the stack. Adaptive Server dynamically allocates and frees memory during the task's runtime. The heap memory pool dramatically reduces the predeclared stack size for each task, while also improving the efficiency of memory usage in the server. The heap memory the task uses is returned to the heap memory pool when the task is finished.

Set the heap memory with the **heap memory per user** configuration parameter. The syntax is:

```
sp_configure 'heap memory per user', amount_of_memory
```

Heap memory is measured in bytes per user. By default, the amount of memory is set to 4096 bytes. This example specifies setting the default amount of heap memory for 10 users:

```
sp_configure 'heap memory per user', 4096
```

You can also specify the amount of memory in the number of bytes per user. For example, the following example specifies that each user connection is allocated 4K bytes of heap memory:

```
sp_configure 'heap memory per user', 0, "4K"
```

At the initial Adaptive Server configuration, 1MB is set aside for heap memory. Additional heap memory is allocated for all the user connections and worker processes for which the server is configured, so the following configuration parameters affect the amount of heap memory available when the server starts:

number of user connections

number of worker processes

The global variable @@*heapmemsize* reports the size of the heap memory pool, in bytes.

Calculating heap memory

To calculate how much heap memory Adaptive Server sets aside, perform the following (Adaptive Server reserves a small amount of memory for internal structures, so these numbers vary from site to site):

((1024 * 1024) + (heap memory in bytes)* (**number of user connections** + **number of worker processes**))

The initial value of (1024 * 1024) is the 1MB initial size of the heap memory pool. Adaptive Server reserves a small amount of memory for internal structures.

For example, if your server is configured for:

- **heap memory per user** – 4K
- **number of user connections** – 25 (the default)
- **number of worker processes** – 25 (the default)

@@*heapmemsize* reports 1378304 bytes.

And the estimated value using the formula above, is:((1024 X 1024) + (4 * 1024 * 50)) = 1253376

Now, if you increase the **number of user connections**, the size of the heap memory pool increases accordingly:

sp_configure 'user connections', 100

@@heapmemsize reports 1716224 bytes. The estimated value in this case comes out to be:((1024 * 1024) + (4 * 1024 * (100 + 25)) = 1560576

If your applications were to fail with the following error message:

There is insufficient heap memory to allocate %ld bytes. Please increase configuration parameter 'heap memory per user' or try again when there is less activity on the system.

You can increase the heap memory available to the server by increasing one of:

- **heap memory per user**
- **number of user connections**
- **number of worker processes**

Sybase recommends that you first try to increase the heap memory per user configuration option before you increase number of user connections or number of worker processes. Increasing the number of user connections and number of worker processes first consumes system memory for other resources, which may cause you to increase the server's maximum memory.

The size of the memory pool depends on the number of user connections. Sybase recommends that you set heap memory per user to at least three times the size of your logical page.

How Adaptive Server uses memory

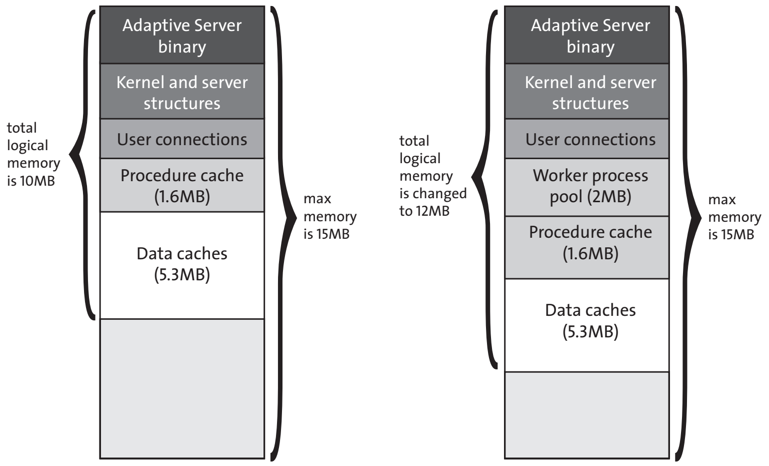
Memory exists in Adaptive Server as total logical or physical memory:

- **Total logical memory** – is the sum of the memory required for all the **sp_configure** parameters. The total logical memory is required to be available, but may or may not be in use at a given moment. The total logical memory value may change due to changes in the configuration parameter values.
- **Total physical memory** – is the sum of all shared memory segments in Adaptive Server. That is, total physical memory is the amount of memory Adaptive Server uses at a given moment. You can verify this value with the read-only configuration parameter **total physical memory**. The value of **total physical memory** can only increase because Adaptive Server does not shrink memory pools once they are allocated. You can decrease the amount of total physical memory by changing the configuration parameters and restarting Adaptive Server.

When Adaptive Server starts, it allocates memory for:

- Memory used by Adaptive Server for non-configurable data structures
- Memory for all user-configurable parameters, including the data cache, the procedure cache, and the default data cache.

Figure below illustrates how Adaptive Server allocates memory as you change some of the memory configuration parameters:



How Adaptive Server handles memory configuration changes

When a 2MB worker process pool is added to the Adaptive Server memory configuration, the procedure and data caches maintain their originally configured sizes; 1.6MB and 5.3MB, respectively. Because **max memory** is 5MB larger than the **total logical memory** size, it easily absorbs the added memory pool. If the new worker process pool brings the size of the server above the limit of **max memory**, any command you issue to increase the worker process pool fails. If this happens, the total logical memory required for the new configuration is indicated in the **sp_configure** failure message. Set the value of **max memory** to a value greater than the **total logical memory** required by the new configuration. Then retry your **sp_configure** request.

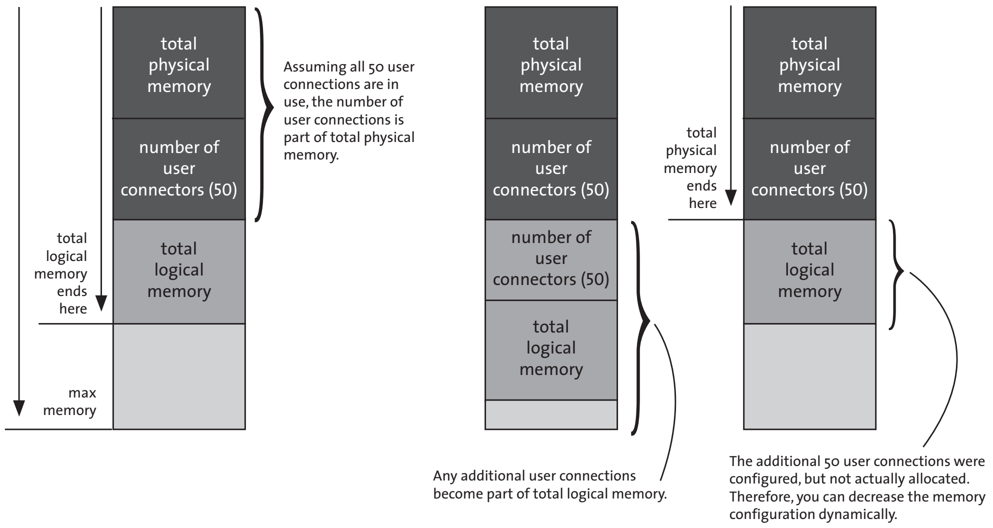
The values for max memory and total logical memory do not include adaptive Server memory

The size of the default data cache and the procedure cache has a significant impact on overall performance.

Dynamically decreasing memory configuration parameters

If you reset memory configuration parameters to a lower value, any engaged memory is not released dynamically. To see how the changes in memory configuration are decreased, see Figure 1 and Figure 2

Figure 1: dynamic allocation on demand set to 1 with no new user connections



total physical memory
number of user connectors (50)
total logical memory
total logical memory ends here
max memory
Assuming all 50 user connections are in use, the number of user connections is part of total physical memory.
total physical memory
number of user connectors (50)
number of user connectors (50)
total logical memory
Any additional user connections become part of total logical memory.
total physical memory ends here
total physical memory
number of user connectors (50)
total logical memory
The additional 50 user connections were configured, but not actually allocated. Therefore, you can decrease the memory configuration dynamically.

In Figure 1, because **dynamic allocation on demand** is set to 1, memory is now used only when there is an event that triggers a need for additional memory use. In this example, such an event would be a request for additional user connections, when a client attempts to log in to Adaptive Server.

You may decrease **number of user connections** to a number that is greater than or equal to the number of user connections actually allocated, because, with **dynamic allocation on demand** set to 1, and without an actual increase in user connection request, no additional memory is required from the server.

Figure 2: dynamic allocation on demand set to 1, with new user connections logged on

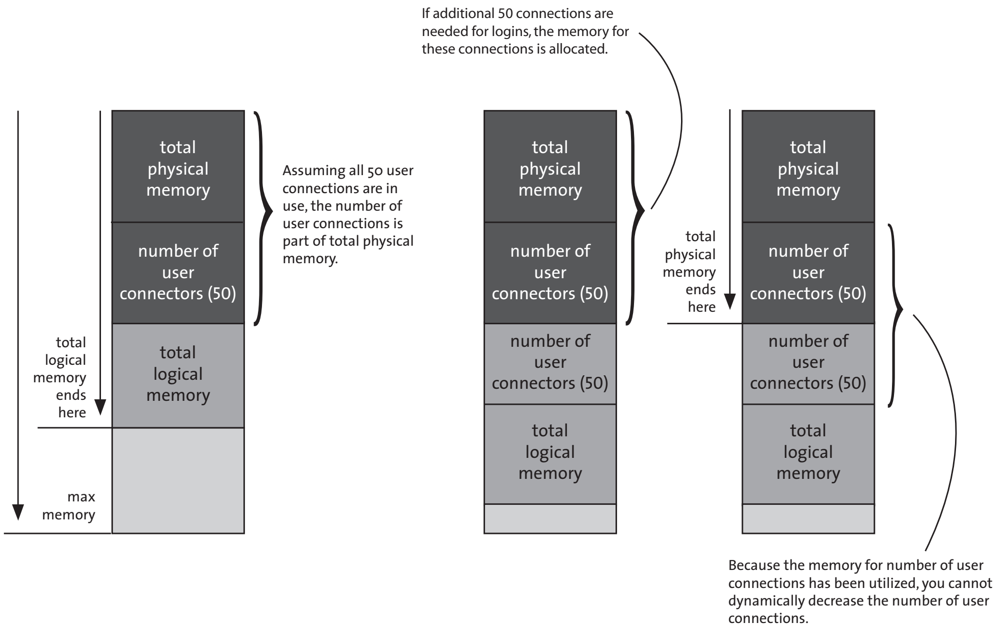
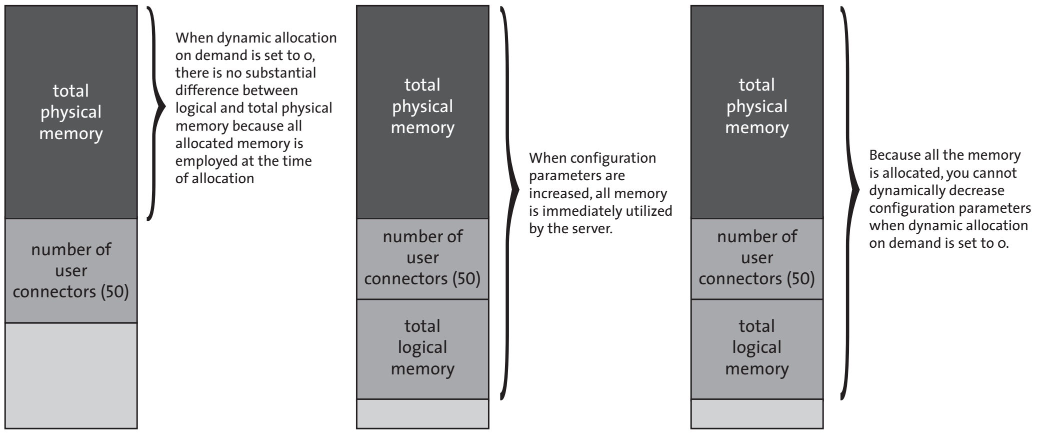

If additional 50 connections are needed for logins, the memory for these connections is allocated.
total physical memory
number of user connectors (50)
total logical memory
Assuming all 50 user connections are in use, the number of user connections is part of total physical memory.
total logical memory ends here
max memory
total physical memory
number of user connectors (50)
number of user connectors (50)
total logical memory
total physical memory ends here
total physical memory
number of user connectors (50)
number of user connectors (50)
total logical memory
Because the memory for number of user connections has been utilized, you cannot dynamically decrease the number of user connections.

Figure 2 assumes that each of the additional 50 user connections is actually used. You cannot decrease **number of user connections**, because the memory is in use. You can use **sp_configure** to specify a change to memory configuration parameters, but this change does not take place until the server is restarted.

Figure 3: dynamic allocation on demand set to 0


total physical memory
number of user connectors (50)
When dynamic allocation on demand is set to 0, there is no substantial difference between logical and total physical memory because all allocated memory is employed at the time of allocation
total physical memory
number of user connectors (50)
total logical memory
When configuration parameters are increased, all memory is immediately utilized by the server.
total physical memory
number of user connectors (50)
total logical memory
Because all the memory is allocated, you cannot dynamically decrease configuration parameters when dynamic allocation on demand is set to 0.

In theory, when **dynamic allocation on demand** is set to 0, there should be no difference between total logical and physical memory. However, there are some discrepancies in the way that Adaptive Server estimates memory needs, and the way in which memory is actually required for the usage. For this reason, you may see a difference between the two during runtime.

When **dynamic allocation on demand** is set to 0, all configured memory requirements are immediately allocated. You cannot dynamically decrease memory configuration.

In Figure2 and Figure 3, users can change the memory configuration parameter values to any smaller, valid value. While this change does not take place dynamically, it disallows new memory use. For example, if you have configured **number of user connections** to allow for 100 user connections and then change that value to 50 user connections, in the situations represented by Figure 2and Figure 3, you can decrease the **number of user connections** value back to 50. This change does not affect the memory used by Adaptive Server until after the server is restarted, but it prevents any new users from logging in to the server.

3.4 Database Security

MySQL

The security of the MySQL host server is an essential first step in enhancing the overall security of the database server (and thus the data). Without first securing the host server, no amount of MySQL security will keep your data safe. There are many ways of securing the server, some of which are given below:

Run the Server as a non-privileged user

In Linux/Unix, you can set the user account that runs the MySQL server process, *mysqld*. (This is opposed to running the server process as the *root* superuser on the system.)

By running the server as a normal user in Linux/Unix, the server has only the same privilege as that user. In other words, the attacker cannot gain further access or perform other privileged processes on the server.

Firewalling the MySQL Server

The most effective way to protect the MySQL server from outside attack is to prevent people from the outside from gaining access to the server. Installing a firewall to protect the server will help to prevent access from non-trusted hosts.

Using a *passive* firewall is a good solution to simply prevent the attacks. The use of firewall stops outside or non-trusted locations from even knowing that a MySQL server is available at your location.

Another option is to use an *active* firewall that listens on common (and even uncommon) ports for a non-trusted user sending a port scan or probe to your machines. The active firewall then completely blocks access from that IP address to all ports.

Communicating securely with MySQL server

As of version 4 of MySQL, communication over TCP via Secure Sockets Layer (SSL) is possible. With SSL support enabled, the traffic between the MySQL server and the client is encrypted. The SSL option is not enabled by default. However, you can enable it and even require it through the use of the additional user privilege system when SSL is compiled into MySQL.

Using Socket based connections

By default, MySQL listens for connections both through sockets for local connections and via TCP/IP for remote connections. If you will not be connecting to MySQL from any hosts other than MySQL server itself, you should disable the TCP/IP option in MySQL.

You can disable TCP/IP based connections by adding the -- *skip-networking* option to the command line when starting the server. Another way to disable TCP/IP access is to add the *skip networking* option to the [mysqld] section of the MySQL configuration file.

Changing the MySQL default port

Another method is to change the port that MySQL server listens on for connections. By default, MySQL listens on TCP port 3306 for connections. To change the default port that MySQL listens for TC/IP connections on, add the line *port=N* to the, section of the MySQL configuration file.

Monitoring data sent to MySQL

When data is entered into a database from applications, especially those that allow users to type their own values, you must check data for errors and other anomalies. The errors occur from two sources:

- **A malicious attack:** With intentional attacks on the applications and database, the attacker may attempt to escape DDL statements into the application. There are simple methods for preventing this type of attack including a least privileged user and data cleansing.
- **Normal Users:** Sometimes normal users of the application enters illegal values. The developer must account for these errors and provide some feedback to the user via error messages.

Disabling DNS

One method of attacking a server or manipulating data is to masquerade as a trusted server or client in MySQL client-server exchange. This type of attack is possible against all applications that utilize Domain Name System (DNS) data, not just MySQL. For a DNS attack to occur, the attacker must be able to alter DNS data on one of the resolvers for your MySQL server. An attacker could also spoof packets within the communication. If DNS data is altered or untrusted an attacker could also pretend to be a trusted host because the host is part of the MySQL authentication scheme. A DNS attack could also be performed against the client also. In a client attack, false information is passed to the client and a fake MySQL server poses as a real server (thus getting the authentication information and data being passed from client to the server).

To prevent a DNS attack from being successful, you can turn off hostname lookups in MySQL server. All connections will be based on IP address, with the exception of the local hostname connections. To disable DNS use from MySQL, start *mysqld* with the command-line switch *- -skip-name-resolve*. You can also add *skip-name-resolve* to the MySQL configuration file.

MySQL Authentication and Privileges

The MySQL privilege system works on a number of levels prior to allowing access to the server. Unlike systems where simply a username and password are examined to determine access, MySQL uses the username, password, and host to determine access level for the database.

MySQL uses a two-stage process for determination of your access level, the connection level and process level. Using these two levels, in two sequential steps, the MySQL server determines

- Whether you are allowed to connect at all.
- Whether you have privilege to perform the requested operation or statement.

During the first stage, which is Connection Stage, the MySQL server combines the user and host provided as credentials and determines if the given combination is allowed to connect with the given password.

1. The first phase of the Connection Stage combines the *host*, *user*, and *password* columns from the *users* table of the MySQL *grants* database.
   - If no database is included in the connection request, access is granted or denied at this point. The host column within the grants database can contain any valid hostname, IP address, or localhost. In addition, the wildcards % and _ are valid as are netmask values. For all the hosts, the % wildcard can be used. For example, *username* '192.168.1. %' would grant access to username from any address within the 192.1.168.0/24 range.
   - If a database is included within the connection request, the second phase begins.
2. The second phase of the Connection Stage of authentication is to verify credentials of the database.

   This phase is performed against the *db* table of the MySQL *grants* database. The *db* table is accessed for host, database, and user. If access to all the databases is granted to the user, this stage automatically passes; otherwise access is denied or granted, depending on the information in the *db* table.
3. The third phase of the Connection Stage is performed against the *host* table of the MySQL *grants* database.

   If connections from host are restricted in some way, this table determines appropriate access.

During the next (second) stage of MySQL authentication, the requested process is examined for specific privileges to determine access. For example, if the user attempts to issue a SELECT statement, the authentication process looks again at the *user* table of the MySQL *grants* database. If authentication passes the *user* table, it is again passed on to the *db* table and then to the host table. If the statement is run against a table, the *table_priv* table is also consulted for authentication, if the statement is run against a column, then the *column_priv* table is consulted.

MySQL Privileges

Each table of the MySQL *grants* database provides the privileges shown in table below (with the exception of RELOAD, SHUTDOWN, PROCESS, and FILE privileges, which are limited to the user table because they have no meaning in other contexts):

Privilege	**Function**
SELECT	Table-level privilege for selecting data or performing queries.
INSERT	Table-level privilege for adding data.
UPDATE	Table-level privilege for updating or changing data.
DELETE	Table-level privilege for deleting data from tables.
INDEX	Table-level privilege for creating and deleting indexes.
ALTER	Table-level privilege for changing table layout.
CREATE	Database-, Table-, and index- level privilege for creation of databases, tables and indexes.
DROP	Database- and Table-level privilege for deleting databases and tables.
GRANT	Database- and Table-level privilege for enabling a user to alter privileges for other users including adding and deleting users.
REFERENCES	Database- and Table-level privilege for using references.
RELOAD	Server- level privilege for reloading and flushing server parameters.
SHUTDOWN	Server- level privilege for stopping MySQL database server.
PROCESS	Server- level privilege to enable process listing and killing.
FILE	Server- level privilege to work with files such as selecting into outfiles.

MySQL Passwords

MySQL passwords are stored with an encryption algorithm. MySQL includes a function to create a valid MySQL encryption: PASSWORD (). Using the PASSWORD function, a user with access to a MySQL CLI could create a valid encryption for use in MySQL. The user could then take the encryption and transfer it to the server administrator for addition to the MySQL grants database.

Adding users and granting privileges

In MySQL, adding a user and granting privileges can occur simultaneously. You can also add multiple users- with the same privileges- simultaneously. The basic statement for adding a user and setting privileges is GRANT statement. The syntax of GRANT statement is as follows:

```
GRANT privilege [ (<columnlist>)] [ , privilege [ (<columnlist>)] …]
    ON { <tablename>  |*|*. *| <databasename>. *}
    TO username  (@<host>) [ IDENTIFIED BY  'password' ] …]
    [ REQUIRE
           [ { SSL| X509} ]
           [ CIPHER cipher [ AND] ]
           [ ISSUER issuer  [ AND] ]
           [ SUBJECT subject] ]
    [ WITH GRANT OPTION]
```

To use the GRANT option, you must have GRANT privilege.

With the GRANT statement you can also specify that the privilege will only apply to certain columns within a given table. You can also specify more than one privilege within a statement and apply that to the same or different columns within the same table or *database. table* structure. In addition, you can give the same privilege to multiple users at the same time if you separate the *users/host/password* portions with the commas.

If you are unsure of the grants that a given user has, you can issue the SHOW GRANTS FOR statement.

For example,

```
SHOW GRANTS FOR Webuser@localhost;
```

Deleting users and revoking privileges

The REVOKE statement is used to revoke privileges from a user. The syntax for the REVOKE statement is as follows:

```
REVOKE privilege [(<columnlist>)]. [, privilege [(<columnlist>)]…]
       ON [<tablename> | * | *. * | <databasename>. *]
       FROM username [, username …]
```

The ALL PRIVILEGES macro works with the REVOKE statement the same as with the GRANT statement. This is important to know because if you have granted the GRANT privilege to the user and use a REVOKE ALL PRIVILEGES statement, the GRANT option will still be there! Therefore, you must perform a separate statement of REVOKE GRANT OPTION… for this occasion.

Issuing a REVOKE statement does not delete the user from the MySQL grants database. To delete a user from the grants database you must specifically issue a DELETE statement to remove them from the grants database.

Remember that MySQL uses the user and host combined to identify a user. Therefore you can have as many username Robert in the database, but only one Robert that connects from a specific host. Therefore you must be extremely careful while issuing the DELETE statement so as to not delete all those other Robert from the database. A misguided or careless DELETE statement can result in removal of all privileges for all users, not just for Robert. After issuing the correct DELETE statement, don't forget to run the FLUSH PRIVILEGES statement so as the deletion takes effect on the server.

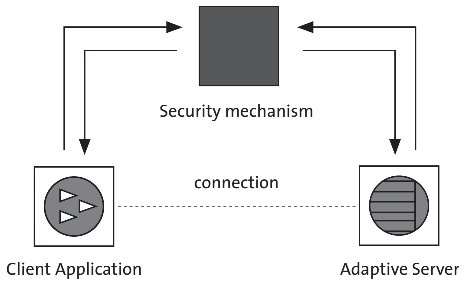
Revoking all privileges from the user won't necessarily truly remove all of their privileges. Specific privileges needs to be specifically removed. The SHOW GRANTS FOR statement can be quite useful in this situation to determine what specific privileges a user has- for example, SHOW GRANTS FOR Webuser@localhost.

Sybase ASE

How applications use security services

The following illustration shows a client application using a security mechanism to ensure a secure connection with Adaptive Server.

Establishing secure connections between a client and Adaptive Server



The secure connection between a client and a server can be used for:

- Login authentication
- Message protection

1. Login authentication

If a client requests authentication services:

1. The client validates the login with the security mechanism. The security mechanism returns a credential, which contains security-relevant information.
2. The client sends the credential to Adaptive Server.
3. Adaptive Server authenticates the client's credential with the security mechanism. If the credential is valid, a secure connection is established between the client and Adaptive Server.

2. Message protection

If the client requests message protection services:

1. The client uses the security mechanism to prepare the data packet it sends to Adaptive Server. Depending upon which security services are requested, the security mechanism might encrypt the data or create a cryptographic signature associated with the data.
2. The client sends the data packet to Adaptive Server.
3. When Adaptive Server receives the data packet, it uses the security mechanism to perform any required decryption and validation.
4. Adaptive Server returns results to the client, using the security mechanism to perform the security functions that were requested; for example, Adaptive Server may return the results in encrypted form.

Adaptive Server supports seven security functions:

- **Security audit** – an 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.
- **User data protection** – Adaptive Server implements the discretionary access control policy over applicable database objects: databases, tables, views, and stored procedures.
- **Identification and authentication** – Adaptive Server provides its own identification and authentication mechanism in addition to the underlying operating system mechanism.
- **Security management** – functions that allow you to manage users and associated privileges, access permissions, and other security functions such as the audit trail. These functions are restricted based on discretionary access control policy rules, including role restrictions.

- **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.
- **Resource utilization** – Adaptive Server provides resource limits to prevent queries and transactions from monopolizing server resources.
- **TOE access** – Adaptive Server allows authorized administrators to construct login triggers that restrict logins to a specific number of sessions and restrict access based on time. Authorized administrators can also restrict access based on user identities.

Identification and authentication

Adaptive Server uses the Server User Identity (SUID) to uniquely identify a user with a login account name. This identity is linked to a particular User Identity (UID) in each database. Access controls use the identity when determining whether to allow access for the user with this SUID to an object. Authentication verifies that a user is actually the person they claim to be. Adaptive Server allows both internal and external mechanisms for authentication.

External authentication

Security is often enhanced in large, heterogeneous applications by authenticating logins with a central repository. Adaptive Server supports a variety of external authentication methods:

- **Kerberos** – provides a centralized and secure authentication mechanism in enterprise environments that include the Kerberos infrastructure. Authentication occurs with a trusted, third-party server calls a Key Distribution Center (KDC) to verify both the client and the server.
- **LDAP User Authentication** – Lightweight Directory Access Protocol (LDAP) provides a centralized authentication mechanism based on a user's login name and password.
- **PAM User Authentication** – Pluggable Authentication Module (PAM) provides a centralized authentication mechanism that uses interfaces provided by the operating system for both administration and runtime application interfaces.

  In a distributed client/server computing environment, intruders can view or tamper with confidential data. Adaptive Server works with third-party providers to provide security services that:

- Authenticate users, clients, and servers – make sure they are who they say they are.
- Provide data confidentiality with encryption – ensure that data cannot be read by an intruder.
- Provide data integrity – prevent data tampering, and detect when it has occurred.

Policy-Based Access Control

The policy-based access control provides a powerful and flexible means of protecting data, down to the row level. Administrators define security policies that are based on the value of individual data elements, and the server enforces these policies transparently. Once an administrator defines a policy, it is automatically invoked whenever the affected data is queried through applications, ad hoc queries, stored procedures, views, and so on.

Using the policy-based access control simplifies both the security administration of an Adaptive Server installation and the application development process because it is the server, not the application that enforces security. This allows developers to concentrate on implementing business functionality while administrators focus on defining a security policy to enforce consistently across the entire server. These are the features that allow you to implement policy-based access control:

- Access Rules
- Application Context Facility
- Login Triggers
- Domain Integrity Rules

Division of roles

An important feature in Adaptive Server is the division of roles. The roles supported by Adaptive Server enable you to enforce and maintain individual accountability. Adaptive Server provides system roles, such as System Administrator and System Security Officer, and user-defined roles, which are created by a System Security Officer.

Roles provide individual accountability for users performing operational and administrative tasks. Their actions can be audited and attributed to them.

A System Security Officer can define role hierarchies such that if a user has one role, the user automatically has roles lower in the hierarchy. For example, the "chief_financial_officer" role might contain both the "financial_analyst" and the "salary_administrator" roles. The Chief Financial Officer can perform all tasks and see all data that can be viewed by the Salary Administrators and Financial Analysts.

Two roles can be defined to be mutually exclusive for:

- **Membership** – a single user cannot be granted both roles. For example, an installation might not want a single user to have both the "payment_requestor" and "payment_approver" roles to be granted to the same user.
- **Activation** – a single user cannot activate, or enable, both roles. For example, a user might be granted both the "senior_auditor" and the "equipment_buyer" roles, but the installation may not want to permit the user to have both roles enabled at the same time.

System roles, as well as user-defined roles, can be defined to be in a role hierarchy or to be mutually exclusive. For example, you might want a "super_user" role to contain the System Administrator, Operator, and Tech Support roles. In addition, you might want to define the System Administrator and System Security Officer roles to be mutually exclusive for membership; that is, a single user cannot be granted both roles.

3.5 Process Management

MySQL

Understanding how to manage MySQL cluster requires knowledge of four essential processes.

- *mysqld* is a traditional MySQL server process. To be used with MySQL Cluster, *mysqld* needs to be built with support for the NDB Cluster storage engine, as it is in the precompiled *-max* binaries available from **http://dev.mysql.com/downloads/**.
- *ndbd* is the process that is used to handle all the data in tables using the NDB cluster storage engine. This is the process that empowers a storage node to accomplish distributed transaction handling, node recovery, checkpointing to disk, online backup, and related tasks.
- The management server (*ndb_mgmd*) is the process that reads the cluster configuration file and distributes this information to all nodes in the cluster that request it. It also maintains a log of cluster activities. Management clients can connect to management server and check the cluster's status.
- The management client (*ndb_mgm*) process is actually not needed to run the cluster. Its value lies in providing a set of commands for checking the cluster's status, starting backups, and performing other administrative functions.

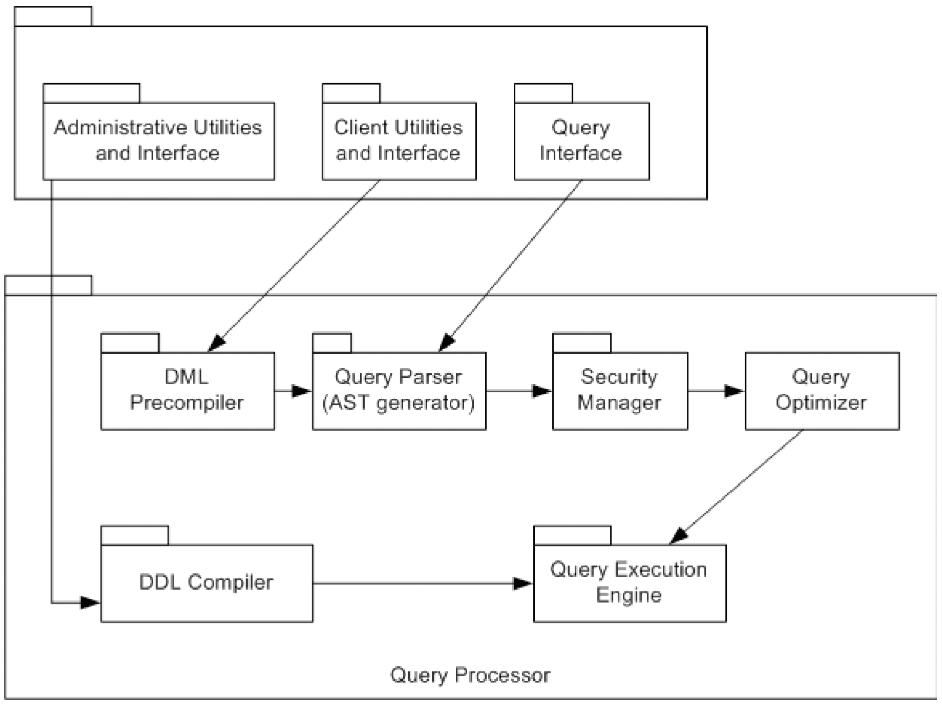
Sybase ASE

Sybase 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_addexeclass, and sp_bindexeclass.

3.6 Query Processor

MySQL

The Query Processor is one of the components in the logical layer. Most of the interactions in the system occur when a user tries to view, manipulate or add to the underlying data in the storage. These interactions are triggered through queries specified by the user using SQL. The queries are parsed and optimized by the query processor.



DML Precompiler

This component forms the very first step in the processing of client applications/commands written in some programming language like C, C++ or Perl. On getting a client request from the Applications Layer, the DML precompiler extracts the relevant SQL statements embedded in the client commands. i.e. translates the client commands to corresponding SQL statements.

DDL Compiler

The DDL compiler handles all requests (mainly from system administrators) for accessing the MySQL databases. This compiler compiles the commands (SQL statements) to interact directly with the database. This is done this way so that administrative utilities do not expose an interface and therefore the DML precompiler does not process such requests.

Query Parser/AST Generator

After the relevant SQL statements are deciphered from the client/administrator request, they are parsed to check for correct syntax and stored in an Abstract Syntax Tree (AST) structure, which is easily understood by the other components in the pipeline. This AST is used to do syntax and semantic checks of the SQL query for validation. In case of an incorrect query, the client is notified of the error.

Security Manager

A query deemed valid is checked for the access control lists of the client. This is done by the security manager. This component verifies the authentication of the client in accessing/connecting to the particular database in question and the client's record and tables privileges. It therefore prevents malicious users from accessing the database.

Query Optimizer

The query optimizer prepares the most efficient plan of query execution by streamlining the query for use by the Query Execution Engine. The optimizer checks to find the index to be used for the quickest and efficient data retrieval. It basically selects the optimal logical plan.

Query Execution Engine

The query execution engine executes the plan it receives from the optimizer. It interprets the plan based on the received information by making requests to the other components in the logical and physical layer (buffer manager, transaction manager etc.) to retrieve the records. The execution engine performs other tasks such as repair, recovery, copying and backup corresponding to requests it receives from the DDL compiler.

Sybase ASE

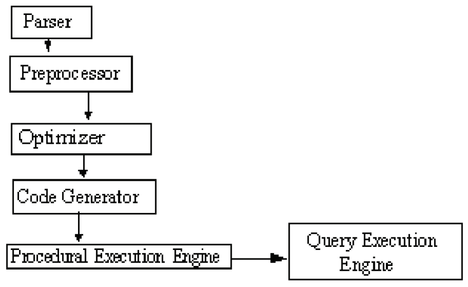
The query processor is designed to process queries you specify. The processor yields highly efficient query plans that execute using minimal resources and ensure that results are consistent and correct.

The query processor uses this information to process a query efficiently:

- The specified query
- Statistics about the tables, indexes, and columns named in the query
- Configurable variables

The query processor has to execute several steps to successfully process a query.

Query Processor modules



- The parser converts the text of the SQL statement to an internal representation called a query tree.
- The preprocessor transforms the query tree for some types of SQL statements, such as SQL statements with sub queries and views, to a more efficient query tree.
- The optimizer analyzes the possible combinations of operations (join ordering, access and join methods, parallelism) to execute the SQL statement, and selects an efficient one based on the cost estimates of the alternatives.
- The code generator converts the query plan generated by the optimizer into a format more suitable for the query execution engine.
- The procedural engine executes command statements such as **create table**, **execute procedure**, and **declare cursor** directly. For Data Manipulation Language (DML) statements, such as **select**, **insert**, **delete**, and **update**, the engine sets up the execution environment for all query plans and calls the query execution engine.
- The query execution engine executes the ordered steps specified in the query plan provided by the code generator.

3.7 Distributed Transactions

MySQL

ndbd is the process that is used to handle all the data in tables using the NDB cluster storage engine. This is the process that empowers a data node to accomplish distributed transaction handling, node recovery, checkpointing to disk, online backups, and related tasks.

In MySQL cluster, a set of ndbd processes cooperates in handling data. These processes can execute on the same computer (host) or on different computers. The correspondences between data nodes and Cluster hosts are completely configurable.

Sybase ASE

In ASE the distributed transactions are handled by distributed transaction manager (DTM). Distributed Transaction Management within Adaptive Server tracks the state of a transaction in the local Adaptive Server/Component Integration Services, as well as in all remote servers participating in transactions. When a user application commits a transaction, the commit is propagated to all participating remote servers using Adaptive Server Transaction Coordinator (ASTC). The management of multisite transactions is handled by ASTC in cooperation with Component Integration Services. Component Integration Services registers new participating servers for each transaction, then turns over control of the transaction coordination to ASTC, which calls back into Component Integration Services to execute various commands for transaction management.

3.8 Transaction Processing

MySQL supports the use of several underlying database engines. Not all engines support explicit management of transaction processing. The two most commonly used engines are **MyISAM** and **InnoDB**. The former does not support explicit transaction management and the latter does. Transaction management is used to maintain database integrity by ensuring that batches of MySQL operations executes completely or not at all.

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_procxmode. 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
```

3.9 Parallel Execution

MySQL

With the current version of MySQL there are two threads used by a slave to implement replication. One reads from the binary log of the master and the other one executes the SQL. If you have a master which is being pounded with individual inserts that are happening on parallel threads these will be serialized by the slave and executed one by one.

Sybase ASE

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.

3.10 Platform Portability

MySQL is available for Windows, LINUX, UNIX, Novell, Sun Solaris, HP-UX, IBM-AIX, Apple Mac OS X, and SCO Open Server. 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.

3.11 Backup and Restore

MySQL

Backup

The easiest way to **backup** your database would be to telnet to your database server machine and use the *mysqldump* command to dump your whole database to a backup file. By default, the output of the command will dump the contents of the database in SQL statements to your console. This output can then be piped or redirected to any location you want.

You can use *mysqldump* to create a simple backup of your database using the following syntax:

```
mysqldump –u [username]   -p [password] [databasename] >
        [backupfile.sql]
```

- [username] – this is your database username
- [password] – this is the password for your database
- [databasename] – the name of your database
- [backupfile.sql] – the file to which the backup should be written.

The resultant dump file will contain all the SQL statements needed to create the table and populate the table in a new database server. To backup your database 'Customers' with the username 'sadmin' and password 'pass21' to a file custback.sql, you would issue the command:

```
mysqldump -u sadmin -p pass21 Customers > custback.sql
```

You can also ask mysqldump to add a drop table command before every create command by using the option --add-drop-table. This option is useful if you would like to create a backup file which can rewrite an existing database without having to delete the older database manually first.

```
mysqldump --add-drop-table -u sadmin -p pass21 Customers > custback.sql
```

Restore

If you have to re-build your database from scratch, you can easily restore the *mysqldump* file by using the *mysql* command. This method is usually used to recreate or rebuild the database from scratch.

Here's how you would restore your custback.sql file to the Customers database.

```
mysql -u sadmin -p pass21 Customers < custback.sql
```

Here's the general format you would follow:

```
mysql -u [username] -p [password] [database_to_restore] < [backupfile]
```

Now how about those zipped files? You can restore your zipped backup files by first uncompressing its contents and then sending it to mysql.

```
gunzip < custback.sql.sql.gz | mysql -u sadmin -p pass21 Customers
```

You can also combine two or more backup files to restore at the same time, using the *cat* command. Here's how you can do that.

```
cat backup1.sql backup.sql | mysql -u sadmin -p pass21
```

Sybase ASE

ASE uses DUMP and LOAD commands to do backup and restore. 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). 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.

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 *sybsystemprocs*.
- Recovers *model*.
- Creates *tempdb* (by copying *model*).
- Recovers *sybsystemdb*.
- 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.

3.12 Basic Tuning

MySQL

When tuning a MySQL Server, the two most important variables to configure are *key_buffer_size* and *table_cache*.

The following example indicates some typical variable values for different runtime configurations.

- If you have at least 256MB of memory and many tables and want maximum performance with a moderate number of clients, you should use:

```
shell> mysqld_safe --key_buffer_size=64M --table_cache=256 \
              --sort_buffer_size=4M --read_buffer_size=1M &
```

- If you have only 128MB of memory and only a few tables, but you still do a lot of sorting, you can use:

```
shell> mysqld_safe --key_buffer_size=16M --sort_buffer_size=1M
```

- With little memory and lots of connections, you can use:

```
shell> mysqld_safe --key_buffer_size=512K --sort_buffer_size=100K \
                --read_buffer_size=100K &
```

Or even this:

```
shell> mysqld_safe --key_buffer_size=512K --sort_buffer_size=16K \
                --table_cache=32 --read_buffer_size=8K \
                  --net_buffer_length=1K &
```

If you are performing GROUP BY or ORDER BY operations on tables that are much larger than your available memory, you should increase the value of *read_rnd_buffer_size* to speed up the reading of rows following sorting operations.

Sybase ASE

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.
- 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.
- 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)"
```

3.13 Other Administrative 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. 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

4.1 Storage Management and Database Creation

MySQL supports several storage engines that act as handlers for different table types. MySQL storage engines include both those that handle transaction-safe tables and those that handle non-transaction-safe tables. The various storage engines supported by MySQL are MyISAM, MEMORY, EXAMPLE, InnoDB and BDB.

Sybase 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 segments within ASE databases. ASE cannot increase the size of its database devices automatically, but they can be manually resized larger at a later time. ASE has its own built-in mirroring commands for environments without RAID or Logical Volume Management (LVM).

4.2 Database Transaction Log

Databases store data to be accessed and processed by applications. Those applications insert, read, update, or delete data. Each of these activities is performed within a transaction, defined as "a recoverable sequence of operations within an application process". A transaction, also referred to as a "unit of work," does not affect the database unless that transaction is committed.

Combining database operations into transactions is half of the solution to ensure data consistency. The other half is a database manager implementation called *write-ahead logging*. Transactions are logged while they occur, regardless of whether or not the transactions commit. Transactions go from the *log buffer* to *log files* (transactional logging) before any data is written from the buffer pools to the database structures. The files used to log the transactions are called the *transaction logs*.

MySQL can create a number of different log files that make it easy to see what is going on inside *mysqld*:

Log Type	Information Written to Log
The error log	Problems encountered starting, running, or stopping **mysqld**.
The general query log	Established client connections and statements received from clients.
The binary log	All statements that change data.
The slow query log	All queries that took more than *long_query_time* seconds to execute or didn't use indexes.

By default, all log files are created in **mysqld** data directory.

In **Sybase 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. Transactions are written to the data cache, where they advance to the transaction log, and database device. When a rollback occurs, pages are discarded from the data cache. The transaction logs are used to restore data in event of a hardware failure. A checkpoint operation flushes all updated (committed) memory pages to their respective tables.

Transaction logging is required for all databases; only image (blob) fields may be exempt.

During an update transaction, the data page(s) containing the row(s) are locked. This will cause contention if the transaction is not efficiently written. Record locking can be turned on in certain cases, but this requires sizing the table structure with respect to the page size.

4.3 Database Server Configuration

Sybase ASE uses the sp_configure system procedure to specify the server wide options. ASE parameters are dynamic, meaning they take effect immediately after executing sp_configure. ASE stores 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. 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.

4.5 Data Import and Export

MySQL

EMS Data Import for MySQL is a powerful tool to import your data quickly from MS Excel, MS Access, DBF, XML, TXT and CSV files to MySQL tables.

EMS Data Export for MySQL is a powerful program to export your data quickly from MySQL databases to any of 15 available formats, including MS Access, MS Excel, MS Word (RTF), HTML, XML, PDF, TXT, CSV, DBF and more.

Sybase ASE

The bcp (Bulk Copy) utility provided by Sybase ASE is used for exporting the data using a native binary format (the -n option) or human-readable character format (the -c option).

ASE provides direct data import/export using CIS and the services of Enterprise Connect / Data Access (ECDA).

4.6 Job Scheduling and Alerts

MySQL Events are tasks that run according to a schedule. Therefore, sometimes they are referred as *scheduled* events. When you create an event, you are creating a named database object containing one or more SQL statements to be executed at one or more regular intervals, beginning and ending at a specific date and time.

Events are executed by a special *event scheduler thread*. When running, the event scheduler thread and its current state can be seen by users having the SUPER privilege in the output of SHOW PROCESSLIST.

The global variable *event_scheduler* determines whether the Event Scheduler is enabled and running on the server. When the server is running event_scheduler can be toggled between ON and OFF (using SET). It is also possible to use 0 for OFF, and 1 for ON when setting this variable.

Job Scheduler of **SYBASE 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 command line 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

Sybase 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

Adaptive server stores the 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

MySQL uses Informant and Incache tools for auditing database and monitoring database security. *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

Sybase ASE has number of trace flags that are used to alter the behavior of the server in certain ways. 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 all the products. Sybase Customer Service and Support can provide advice on the use of particular trace flags.

4.11 System Stored Procedures

One of the most important Transact-SQL extensions is the ability to create stored procedures. A stored procedure is a collection of SQL statements and optional control-of-flow statements stored under a name. The creator of a stored procedure can also define parameters to be supplied when the stored procedure is executed.

Stored routines (procedures and functions) are supported in **MySQL** 5.0. MySQL follows the SQL: 2003 syntax for stored routines. The MySQL implementation of stored routines is still in progress.

The ability to write your own stored procedures greatly enhances the power, efficiency, and flexibility of the SQL database language. Since the execution plan is saved after stored procedures are run, stored procedures can subsequently run much faster than standalone statements. Stored procedures supplied by **Adaptive Server**, called system procedures, aid in Adaptive Server system administration. You can execute stored procedures on remote servers. All Transact-SQL extensions support return values from stored procedures, user-defined return status from stored procedures, and the ability to pass parameters from a procedure to its caller.

4.12 Character Set Support

MySQL includes character set support that enables you to store data using a variety of character sets and perform comparisons according to a variety of collations. A commonly used MySQL character encoding is UTF-8.

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 The DBCC Command

The Database Consistency Checker (dbcc) is a set of utility commands for checking the logical and physical consistency of a database. User ids that have been granted the SA_role or the dbo are automatically granted permission to use all the dbcc commands. The *checktable* and *checkdb* commands check table(s) to see that:

- Index and data pages are correctly linked
- Indexes are in properly sorted order
- All pointers are consistent
- The data rows on each page all have entries in the first Object Allocation Map (OAM) page matching their respective locations on the page

The tablealloc, indexalloc, and checkalloc commands check object to see that:

- All pages are correctly allocated
- No page is allocated that is not used
- No page is used that is not allocated

Correcting allocation errors using the fix option with user tables, will correct any allocation errors found while the dbcc command is executing.

The dbcc checkdb, checktable, checkalloc, tablealloc, and indexalloc commands can be run while the database is active. A shared lock will be put on the object while the above dbcc commands are executing with the exception of the checkalloc option. The checkalloc option is performed with no locking.

5 MIGRATING THE DATABASE ARCHITECTURE

Migration of MySQL database to Adaptive Server Enterprise can be divided into three steps:

1. Create the Sybase ASE server and database schema. This include all the database objects, the users, roles and security:
   - Create, configure and determine devices for the Sybase ASE server and database
   - Create the database, set the database options, create the server logins and set the database ownership
   - Create the database schema from the MySQL database and transform into Sybase ASE database schema, run the Sybase ASE schema
2. Create the administration and security
   - describe the MySQL security model
   - translate this into Sybase ASE syntax
3. Migration of data
   - export the MySQL data
   - import into Sybase ASE

This section describes the first two steps and methods to perform successful migration. The first task in the migration of the database is to create an instance of Adaptive server enterprise, which will provide the container for creating database objects and users. The main focus when creating the database should be on its architecture to derive performance that is equivalent to or better than performance obtained by the MySQL database.

5.1 Build the Sybase Instance

This section describes the planning and preparation for installation of Adaptive Enterprise Server Instance.

Pre installing planning

Following guidelines will help to install and configure the ASE15.0:

- Create a "sybase" account on your system to perform all installation tasks. The "sybase" user must have permission privileges from the top (or root) of the disk partition or operating system directory down to the specific physical device or operating system file.
- Maintain consistent ownership and privileges for all files and directories. A single user—the Sybase System Administrator with read, write, and execute permissions—should perform all installation, upgrade, and setup tasks.
- Make sure there is sufficient disk space available where the Adaptive Server software will be installed. There cannot be any spaces in the path name of the directory.
- Verify that the operating system meets the version-level, RAM, and network protocol requirements for your platform.
- For Adaptive Server to run, the operating system must be configured to allow allocation of a shared memory segment at least as large as the Adaptive Server **total logical memory** configuration parameter.
  After you install Adaptive Server, you can change any configuration parameter, procedure cache, and data cache size. This may require that you increase the value of the configuration parameter *max memory.*
- Depending on the number and types of devices you use for backup (dump) and recovery (load), you may need to adjust the **shared memory segment** parameter in the operating system configuration file to accommodate concurrent Backup Server processes.
- It is extremely important that you understand and plan resource usage in advance. In the case of disk resources, for example, after you initialize and allocate a device to Adaptive Server, that device cannot be used for any other purpose (even if Adaptive Server never fills the device with data). Likewise, Adaptive Server automatically reserves the memory for which it is configured, and this memory cannot be used by any other application.
- For recovery purposes, it is always best to place a database's transaction log on a separate physical device from its data.
- Adaptive Server contains many features that optimize performance for OLTP, decision-support, and mixed workload environments. However, you must determine in advance the requirements of your system's applications to make optimal use of these features.
- Master device and the database can be created with logical page sizes of 2K, 4K, 8K or 16K. It determines the server's space allocation. All the databases in a server and all objects in every database use the same logical page size.

Installation

The following installation options have to be determined before installing the Adaptive Server Enterprise:

- Install directory: A directory for the installation of Adaptive Server Enterprise.
- Type of installation: Three types of installations in the Install Type Window:
  - Typical
  - Full
  - Custom

  License mode: Indicate whether the licenses will be obtained from a license server or will be using unserved licenses.
- Configure server for email notification: When configuration is enabled, designated users will receive information about license management events requiring attention.
- Edition of Adaptive server:
  Choose the editions of Adaptive server from:
  - Enterprise Edition
  - Small Business Edition
  - Developers Edition

- Choose servers to configured: A Full or Custom installation allows you to choose to
  - Configure new Adaptive Server
  - Configure new Backup Server
  - Configure new Monitor Server
  - Configure new XP Server
  - Configure new Job Scheduler
  - Enable Self Management
  - Configure Web Services
  - Configure Enhanced Full-Text Search Server
  - Configure Unified Agent
- If custom configuration is used, then determine following options-
  - Server Name
  - Port Number
  - page size – 2K, 4K, 8K or 16K
  - name of the error log file
  - master device location
  - master device size
  - master database size
  - system procedure device
  - system procedure device size
  - system procedure database size
- The Unified Agent: Choose unified Agent from following:
  - Jini Adapter
  - UDP Adapter (default)

After the successful installation of Adaptive Server instance, following servers will be installed:

- Adaptive Server - <instance_name>
- Backup Server - <instance_name>_BS
- Monitor Server - <instance_name>_MS
- XP Server - <instance_name>_XP
- Job Scheduler - <instance_name>_JSAGENT
- Enhanced full text search - <instance_name>_TEXT

5.2 Configure the Server

ASE Server configuration information is available in the sysconfigures and syscurconfigs table in the master database. A database administrator can modify Server configuration information using the sp_configure stored-procedure followed by executing the RECONFIGURE command to make any changes operational (do not edit the sysconfigures or syscurconfigs table directly). Some Server configuration options are dynamic and come into effect immediately while others require Server to be restarted—these are referred to as static options. Furthermore, Server itself modifies many configuration options to tune itself according to the resources available.

Installation of the Adaptive server includes default parameter settings for all the configuration parameters. A configuration file is created automatically at the time of installation. The default name of the file is *server_name.cfg* and the default location of the file is the Sybase installation directory. A database administrator can edit this file, although any changes made will not go into effect until Sybase is restarted. Alternately, a database administrator can use the **sp_configure** stored- procedure in conjunction with the RECONFIGURE command. When you change a configuration parameter, Adaptive Server saves a copy of the old configuration file as *server_name.001*, *server_name.002*, and so on. Adaptive Server writes the new values to the file *server_name.cfg* or to a file name you specify at start-up. This 'default' setting can be modified depending on the system's need.

Many of the configuration options are dynamic, and using this mechanism has the advantage of allowing dynamic options to go into effect immediately. Configuration parameters are used for a wide range of services, from basic to specific server operations, and for performance tuning.

Server level configuration:

Configuration parameters can be set or changed in one of the following ways:

- By executing **sp_configure** with the appropriate parameters and values,
- By editing your configuration file and then invoking **sp_configure** with the **configuration file** option, or
- By specifying the name of a configuration file at start-up.

Configuration parameters are grouped according to the area of Adaptive Server behavior they affect. This makes it easier to identify all parameters that you might need to tune to improve a particular area of Adaptive Server performance.

Parameter group	**Configures Adaptive Server for:**
Backup/Recovery	Backing up and recovering data
Cache manager	The data and procedure caches
Component Integration Services administration	Component Integration Services
DTM administration	Distributed transaction management (DTM) facilities
Diagnostics	Diagnostic principles
Disk I/O	Disk I/O
Error log	Error log and the logging of Adaptive Server events to the Windows Event Log
Extended stored procedures	Affecting the behavior of extended stored procedures (ESP's).
General information	Basic system administration
Java services	Memory for Java in Adaptive Server See the *Java in Adaptive Server Enterprise* manual for complete information about Java in the database. If you use method calls to JDBC, you may need to increase the size of the execution stack available to the user.
Languages	Languages, sort orders, and character sets
Lock manager	Locks
Memory use	Memory consumption
Meta-data caches	Setting the metadata cache size for frequently used system catalog information. The metadata cache is a reserved area of memory used for tracking information on databases, indexes, or objects. The greater the number of open databases, indexes, or objects, the larger the metadata cache size.
Monitoring	Collecting monitoring information. By default, Adaptive Server does not collect monitoring information required by the monitoring tables.
Network communication	Communication between Adaptive Server and remote servers, and between Adaptive Server and client programs
O/S resources	Use of operating system resources

Parameter group	Configures Adaptive Server for:
Physical memory	Your machine's physical memory resources
Processors	Processors in an SMP environment
Query Tuning	Query optimization
RepAgent thread administration	Replication via Replication Server
SQL Server administration	General Adaptive Server administration
Security related	Security-related features
Unicode	Unicode-related features
User environment	User environments

For e.g. to set the default locking scheme to be used by when creating a new table to "datapages":

```
sp_configure "lock scheme", 0, "datapages"
```

To set the maximum number of worker processes allowed per query, which is called maximum degree of parallelism to 2, execute following command:

```
sp_configure "max parallel degree", 2
```

For more information on setting the parameters of the above group, see System Administrator Guide, Chapter 5: "Setting Configuration Parameters".

System stored procedure **sp_sysmon** can be run before and after using **sp_configure** to adjust configuration parameters. The output gives a basis for performance tuning and the results of configuration changes can be observed.

Languages and charsets cannot be displayed or edited with sp_configure. To change these parameters, the configuration file must be edited manually.

Database Level configuration:

Database options can be used to configure the settings for an entire database. These database options differ from sp_configure parameters, which affect the entire server. These database option controls:

- The behavior of transactions
- Defaults for table columns
- Restrictions to user access
- Performance of recovery and bcp operations
- Log behavior

To list the complete database settings, use sp_dboption system stored procedure. To change the database option, use the command with parameters.

For e.g., to allow null value on a column by default when creating a table, set following database option:

```
use master
go
sp_dboption database_name, "allow nulls by default", true
go
```

To make an option or options take effect for every new database, change the option in the model database.

5.3 Migrate the Storage Architecture

This section describes the physical and logical storage structure of Adaptive server. It will be helpful to in understanding how to configure the storage in Adaptive Server.

Adaptive Server has been designed for the isolation of administrative duties at the database level. The system (catalog and roles) has been divided, with centralized functions that have instance-wide authority under the master database and database-specific functions under the individual databases.

Before creating the database, logical devices need to be initialized for placing the database on them.

Database Device or Logical Device:

Sybase logical devices are created on OS devices (which can be stripped over several disks). These devices are files or portions of a disk that are used to store databases and database objects. You can initialize devices using raw disk partitions or operating system files.

A pool of default database devices can be created, which can be used by all Adaptive Server users for creating databases. Whenever users create (or alter) databases without specifying a database device, new disk space is allocated from the pool of default disk space.

Database devices can be initialized with the **disk init** command, which:

- Maps the specified physical disk device or operating system file to a *database device* name
- Prepares the device for database storage

disk init command divides the database devices into **allocation units**, groups of 256 logical pages. The size of the allocation unit depends on which logical page size server is configured for (2, 4, 8, or 16K). In each allocation unit, the **disk init** command initializes the first page as the allocation page, which contains information about the database (if any) that resides on the allocation unit.

Database:

The database is created on one or more logical devices. One logical device can contain multiple databases.

Databases can be created either through Sybase Central or by using the CREATE DATABASE T-SQL command. Only System Administrator can create user database or it can grant permission to use create database command to a user of a master database. By default, the creator of the database becomes the owner of the database.

The following basic characteristics of the database have to be decided before creating a database:

Database name: A database name must follow the rules for identifiers. A meaningful unique name can be used and need not be the same as the name of the MySQL database being migrated.

Database owner: By default, the user who created the database becomes the database owner (**dbo**). The owner can be changed using **sp_changedbowner.**

Device Name: To place the database on specific database devices, give the names of the database devices where you want it stored. Database can be stored on more than one database device, with a different amount of space on each. If device name is not specified then Adaptive Server puts the database on one or more of the default device.

Transaction log. Every database has at least one transaction log. The 'log on' clause of 'create database' command can be used to place a production database's transaction log on a separate device for better performance.

A System Administrator usually creates the user databases and gives ownership of them to another user after completing some of the initial work. **sp_changedbowner** changes the ownership of a database. The procedure must be executed by the System Administrator in the database where the ownership is to be changed. The syntax is:

```
sp_changedbowner loginame [ , true]
```

The new owner must already have a login name in Adaptive Server, but he or she cannot be a user of the database or have an alias in the database.

The new database initially contains a set of system tables with entries that describe the system tables themselves. The new database inherits all the changes you have made to the *model* database, including:

- The addition of user names.
- The addition of objects.
- The database option settings. Originally, the options are set to off in *model*. If you want all of your databases to inherit particular options, change the options in *model* with **sp_dboption**.

Segments:

Segments within a database are created to describe the database devices that are allocated to the database.

When database is created, Adaptive Server creates three segments:

- **System:** Stores the database's system tables
- **Log segment:** Stores the database's transaction log

Default: Stores all other database objects—unless you create additional segments and store tables or indexes on the new segments by using **create table…on** segment_name or **create index…on** _data = 20, test_index = 20 log on test_log = 10

```
go

sp_addsegment test_seg_data, testdb, test_data
go

sp_dropsegment test_seg_data, testdb, system
go
sp_dropsegment test_seg_data, testdb, default
go
sp_addsegment test_seg_index, testdb, test_index
go
sp_dropsegment test_seg_index, testdb, system
go
sp_dropsegment test_seg_index, testdb, default
go
use testdb
go
create table test_table …. on test_seg_data
go
create index test_table_index on test_table….on test_index
go
```

Improve performance with Object placement:

Adaptive Server allows controlling the placement of databases, tables, and indexes across physical storage devices. This can improve performance by equalizing the reads and writes to disk across many devices and controllers. For example, you can:

- Place a database's data segments on a specific device or devices, storing the database's log on a separate physical device. This way, reads and writes to the database's log do not interfere with data access
- Spread large, heavily used tables across several devices.
- Place specific tables or non-clustered indexes on specific devices. For example, you might place a table on a segment that spans several devices and its non-clustered indexes on a separate segment.
- Place the text and image page chain for a table on a separate device from the table itself. The table stores a pointer to the actual data value in the separate database structure, so each access to a text or image column requires at least two I/Os.
- Distribute tables evenly across partitions on separate physical disks to provide optimum parallel query performance.

For multi-user systems and multi-CPU systems that perform a lot of disk I/O, pay special attention to physical and logical device issues and the distribution of I/O across devices:

- Plan balanced separation of objects across logical and physical devices.
- Use enough physical devices, including disk controllers, to ensure physical bandwidth.
- Use an increased number of logical devices to ensure minimal contention for internal I/O queues.
- Use a number of partitions that will allow parallel scans, to meet query performance goals.

Make use of the ability of create database to perform parallel I/O on as many as six devices at a time, to gain a significant performance leap for creating multi gigabyte databases.

5.4 Migrate the Database Users

This section explains the migration of users of the MySQL database to the Adaptive server enterprise. The privileges can be granted to the users directly or through roles.

A user will have to be created in Adaptive Server in the database in which the schemas objects will be created. The user is then given privileges in the database to create objects and populate the data by assigning the user as database owner.

The migration of the users can be done in two steps:

1.Create user accounts
2.Create Roles and grant privileges

Create User Account:

Get the list of users with privileges on all the objects of the database by querying the MySQL database.

System stored procedure sp_addlogin is used to create login name in the Adaptive server. Login name is not used to give the permissions to access user databases.

System stored procedure sp_adduser is used to create new users to the individual databases for the logins created. User name is used to give permission to the objects in a database. A user should be created in a database only if there are objects in the database that needs to be access.

Users who have access to a database still need permissions to read data, modify data, and use certain commands. These permissions are granted with the grant and **revoke** commands.

Create Roles and privileges:

The final steps in adding database users are assigning them special roles, as required, and granting permissions.

The roles supported by Adaptive Server enable you to enforce individual accountability. Adaptive Server provides system roles, such as System Administrator and System Security Officer, and user-defined roles, which are created by a System Security Officer. Object owners can grant database access as appropriate to each role.

Before you implement user-defined roles, decide:

- The roles you want to create
- The responsibilities for each role
- The position of each in the role hierarchy
- Which roles in the hierarchy will be mutually exclusive
- Whether such exclusivity will be at the membership level or activation level

A user-defined role can be created using following command:

```
create role roll_name [ with passwd "password"]
```

Roles must be active to have the access privileges of each role. Depending on the default set for a role, the role may or may not be active at login. If the role has a password, it will always be inactive at login.

To activate or deactivate a role:

```
set role role_name [ on|off]
```

To activate or deactivate a role that has an attached password, use:

```
set role role_name with passwd "password" [ on|off]
```

Object privileges can be granted directly to the user or though roles. The following grant command is used to grant object access permissions:

```
grant { all [ privileges] | permission_list}
on { table_name [ (column_list)]
| view_name [ (column_list)]
| stored_procedure_name}
to { public | name_list | role_name}
[ with grant option]
```

And following the revoke command is used to revoke object access permissions:

```
revoke [ grant option for]
{ all [ privileges]  | permission_list}
on { table_name [ (column_list)]
| view_name [ (column_list)]
| stored_procedure_name}
from { public | name_list | role_name}
[ cascade]
```

Here,

- **all** or **all privileges** specifies all permissions applicable to the specified object.
- *permission_list* is the list of permissions that you are granting.
- **on** specifies the object for which the permission is being granted or revoked.
- **public** refers to the group "public," which includes all Adaptive Server users.
- *name_list* includes:
  - Group names
  - User names
  - A combination of user and group names, each separated from the next by a comma
- *role_name* is an Adaptive Server system-defined or user-defined role.
- with grant option in a grant statement allows the user(s) specified in *name_list* to grant the specified object access permission(s) to other users.

The roles can be granted to users or other roles. The syntax for that is:

```
grant role role_granted [{ , role_granted} ...]
to grantee [{ , grantee} ...]
```

where:

- *role_granted* – is the role being granted. Specify any number of roles to be granted.
- *grantee* – is the name of the user or role. Specify any number of grantees.

Groups can be created to grant or revoke permission to more than one user in a single statement. If group name is not provided while creating a user, then the user is made a member of a default group 'public'.

6 MIGRATING THE DATA AND SQL

This section describes the data migration methods, migration of database objects, migrating SQL application code and validation of data migrated.

The data migration task can be broken up into the following three subtasks:

1. **Planning:** It is important to understand the various options available for migrating the data, particularly the advantage and limitations of each of the option, evaluate the characteristics of the source data, and evaluate environmental and business constraints.

2. **Execution:** This subtask involves transferring the data using the data migration tool, migrating the MySQL database to Sybase database objects and changing the MySQL specific SQL code in the application code.
3. **Validation:** This subtask accounts for all the data validation and verifies data integrity.

6.1 Factors in Migration

Understand the factors that need to be considered before making the decision for selecting the data migration method. Following factors decides which migration option to be used for data migration:

- Volume of Data: For larger database the use of bcp is recommended for performance perspective.
- Number of objects to be migrated.
- The capacity to create flat files in the source environment affects the choice of migration.
- Server processing capacity: Running bcp on the same server as the database reduces the network overhead of bcp but it consumes more CPU time.

6.2 Options for Migration

The option for data migration to be used will depend on the factors mentioned in the above section. Following options can be used:

6.2.1 PowerDesigner

Using the PowerDesigner tool you will be able to reverse engineer most of the MySQL database objects. These can then be stored into a PowerDesigner Data Model (PDM). From this model you will be able to create the Sybase database object scripts. See Appendix I for the details on the usage of the PowerDesigner tool.

Users, roles and security details will have to be created manually.

For the data migration there are two options. The option used will depend on the amount of data that needs to be migrated.

The use of BCP is recommended for large databases from a performance perspective. If there are lots of exceptions the use of CIS gives a simpler mechanism to insert data from source to target tables.

6.2.2 Sybase Bulk Copy

BCP provides a convenient, high-speed method for transferring data between a database table and an operating system file. **bcp** can read or write files in a wide variety of formats.

Adaptive Server can accept data in any character or binary format, as long as the data file describes either the length of the fields or the **terminators**, the characters that separate columns.

Using MySQL's bcp and Sybase bcp utility, data from MySQL can be migrated to Adaptive Server via flat files:

- Retrieve the data from all the tables of MySQL database into a text file with field or column separator.
- Set following option in Sybase ASE database to allow bulk copying:

```
sp_dboption <dbname>, "select into/bulkcopy", "on", use <dbname>, “checkpoint”
```

- Copy data of each table retrieved in files from the MySQL database, into corresponding ASE tables using bulk copying utility:

```
bcp <database>. <table> in <filename> -e <errorFilename>
```

It will be faster to do BCP without indexes in place.

For more information about bulk copy, see Appendix J

6.2.3 Component Integration Services and ECDA

CIS (Component Integration Services) and Enterprise Connect Data Access (ECDA) provides transparent access to both Sybase and MySQL databases on different servers. Using CIS tables in remote servers can be accessed if they are local.

The content of the remote table can be transferred into a new table by creating proxy tables. Remote tables are mapped to local proxy tables, which holds metadata. Internally, when a query involving remote table is executed, the storage location is determined, and the remote location is accessed so that data can be retrieved.

The datatypes of MySQL table are converted into the specific adaptive Server types automatically when the data is retrieved.

For more information about CIS, see Appendix H

6.2.4 InfoMaker Tool

Sybase's InfoMaker Tool and its Data Pipeline feature can be used to move the data from MySQL directly to Adaptive Server. Create a data pipeline, which, when executed, pipes the data as specified in the definition of the data pipeline.

When you pipe the data from MySQL to Sybase, InfoMaker makes a best guess at the appropriate destination datatypes. You can correct InfoMaker's best guess in your pipeline definition as needed.

The Data Pipeline painter supports the piping of columns of any datatype.

6.2.5 PowerTransfer Tool

PowerTransfer is an extension to PowerDesigner that allows the transfer of data into Adaptive Server. The source database can be any ODBC data source supported by PowerDesigner. PowerTransfer uses the Sybase bulk copy (bcp) mechanism for high-performance inserts into adaptive Server.

6.3 Migrating the database objects and SQL application code

Migrating Database Objects

Following is the high-level view of the mapping of MySQL objects with Sybase Objects:

Features	**MySQL**	**Sybase**
Constraints	Constraints	Constraints
Indexes	B-Tree indexes	B-Tree indexes
Tables	Relational Tables, Temporary Tables	Relational Tables, Temporary Tables
Procedures	SQL statements	Sybase T-SQL statements, SQLJ Stored procedure
Functions	User Defined Functions Native functions	Stored procedure
Triggers	After Triggers, Before Triggers	Login Triggers, After Triggers
Sequence	Not Supported	Supported
Indexed Views	Not Supported	Not Supported

Constraints:

The functionality provided by MySQL and Adaptive Server to define constraints on columns is almost identical.

Following are the common constraints used by MySQL and adaptive Server:

```
NOT NULL
UNIQUE
PRIMARY KEY
REFERENCE KEY
```

The integrity constraints added on the table can be viewed by sp_helpconstraint system stored procedure.

Triggers:

Both MySQL and Adaptive server do not allow multiple triggers on the same table. In Adaptive Server, trigger fires only after the data modification statement has completed and it has checked for any datatype, rule, or integrity constraint violation. The trigger and the statement that fires it are treated as a single transaction that can be rolled back from within the trigger. If the Adaptive Server detects a severe error, the entire transaction is rolled back.

Views:

Both MySQL and Adaptive server do not allow indexed views. Various types of views available in Adaptive Server are:

- Simple views.
- Views with a computed column.
- Views with an aggregate function or built-in function.
- View with a join.
- Views derived from other views.
- Distinct views.
- Views that includes identity column.

Stored Procedures

In order for Sybase to compile a stored procedure successfully, all of the stored procedures dependency must be defined in the database.

Migrating the SQL application code

Migrating the SQL application code consists of changing MySQL specific SQL. The following has to be checked for SQL language differences described above:

- Stored procedures.
- Triggers
- SQL queries.

See Appendix I for the details on the usage of the PowerDesigner tool. Also, refer to appendix E to find out about the SQL language differences and Appendix -G on any Tuning and Performance that may be needed.

6.4 Validate the Data migration

6.4.1 Verify the data transfer

The log files should be checked for failures and errors while transferring the data from MySQL to Adaptive Server. Count the number of rows of all the tables whose data has been transferred from MySQL to Adaptive server. If any disparity found in the count then the log files should be checked for reason of failure of data transfer.

6.4.2 Validate the data Integrity

Database integrity is automatically checked when creating or enabling the constraints after the data transfer has taken place. Lack of primary key and foreign key constrains in the database require through testing of application to verify the data integrity.

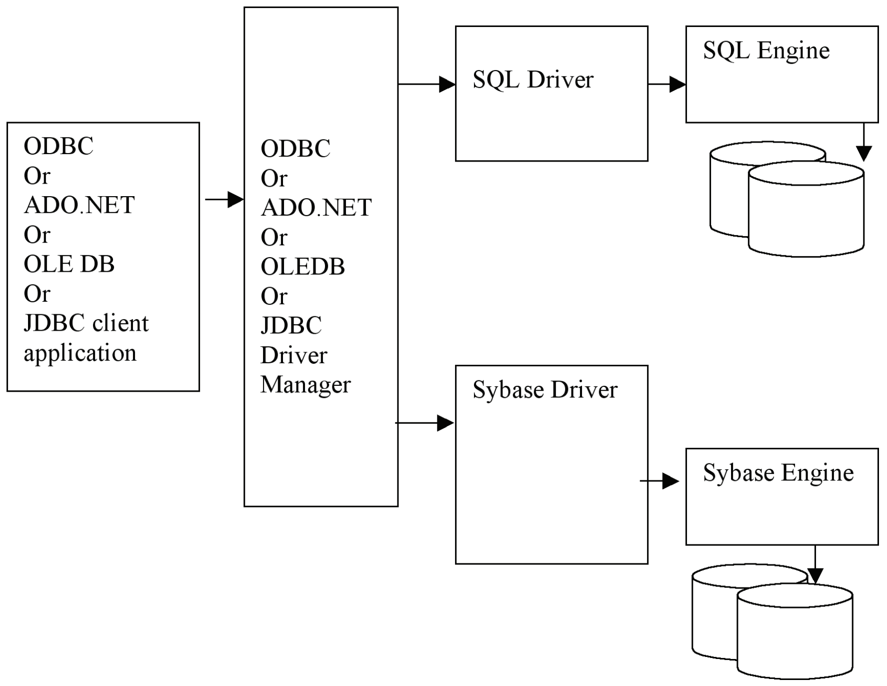
7 MIGRATING CLIENT DATABASE APPLICATION

The data and any SQL code that are stored in the database (e.g., stored procedures and triggers) are migrated with the steps in Section 6. This section describes the different types of client database applications that need to be migrated from MySQL to Sybase.

- ODBC client application
- ADO.NET client application
- OLE DB client application
- JDBC client application

Also, refer to Appendix E to find out about the SQL language differences and appendix G on any Tuning and Performance that may be needed.

It should be fairly straight -forward to migrate an MySQL ODBC client application to a Sybase ODBC client application, or an MySQL JDBC client application to a Sybase JDBC client application. In both cases, the architecture of the application may be:


ODBC Or ADO.NET Or OLE DB Or JDBC client application
ODBC Or ADO.NET Or OLEDB Or JDBC Driver Manager
SQL Driver
SQL Engine
Sybase Driver
Sybase Engine

Sybase has developed its own ODBC Driver and OLE DB Provider for ASE.

7.1 ODBC Client Application

Applications based on ODBC are able to connect to the most of the popular databases. Thus, application conversion is very easy. To convert a MySQL ODBC client application to a corresponding Sybase ODBC client application, the developer needs to change the ODBC client application, and for that follow the steps:

- Install the correct ODBC for Sybase
- Check the version of the ODBC client application and determine if all ODBC statements are supported.
- When an application connects to a database, it uses a set of connection parameters to define the connection. Connection parameters include information such as the server name, the database name, and a user ID. Change these connection parameters to connect to Adaptive Server with correct database.
- Possible changes in calling stored procedures
- Possible logical changes required to retain the program flow

Features supported by ASE ODBC Driver

ODBC features are arranged according to level of conformance. Features are Core, Level 1, or Level 2, with Level 2 being the most complete level of ODBC support.

The ASE ODBC Driver meets Level 2 conformance with the following exceptions:

- **Level 1 conformance** The ASE ODBC Driver supports all Level 1 features, except for asynchronous execution of ODBC functions, **SQLBulkOperations**, **SQLSetPos**, and scrollable cursors.
- **Level 2 conformance** The ASE ODBC Driver supports all Level 2 features, except for asynchronous execution of ODBC functions, using bookmarks.

ASE ODBC Driver works with any ODBC Driver Manager implementation.

ASE ODBC Driver datatype mappings:

ASE datatype	**ODBC SQL type**	**ODBC bind type**
Binary	SQL_BINARY	SQL_C_BINARY
bit	SQL_BIT	SQL_C_BIT
char	SQL_CHAR	SQL_C_CHAR
date	SQL_TYPE_DATE	SQL_C_TYPE_DATE or SQL_C_CHAR
datetime	SQL_TYPE_TIMESTAMP	SQL_C_TYPE_TIMESTAMP or SQL_C_CHAR
decimal	SQL_DECIMAL	SQL_C_NUMERIC or SQL_C_CHAR
double	SQL_DOUBLE	SQL_C_DOUBLE
float (<16)	SQL_REAL	SQL_C_FLOAT
float (>=16)	SQL_DOUBLE	SQL_C_DOUBLE
image	SQL_LONGVARBINARY	SQL_C_BINARY
int [eger]	SQL_INTEGER	SQL_C_LONG
money	SQL_DECIMAL	SQL_C_NUMERIC or SQL_C_CHAR
nchar	SQL_CHAR	SQL_C_CHAR
nvarchar	SQL_VARCHAR	SQL_C_CHAR
numeric	SQL_NUMERIC	SQL_C_NUMERIC or SQL_C_CHAR
real	SQL_REAL	SQL_C_FLOAT

ASE datatype	ODBC SQL type	ODBC bind type
smalldatetime	SQL_TYPE_TIMESTAMP	SQL_C_TYPE_TIMESTAMP or SQL_C_CHAR
smallint	SQL_SMALLINT	SQL_C_SHORT
smallmoney	SQL_DECIMAL	SQL_C_NUMERIC or SQL_C_CHAR
text	SQL_LONGVARCHAR	SQL_C_CHAR
time	SQL_TYPE_TIME	SQL_C_TYPE_TIME or SQL_C_CHAR
timestamp	SQL_BINARY	SQL_C_BINARY
tinyint	SQL_TINYINT	SQL_C_TINYINT
unichar	SQL_WCHAR	SQL_C_CHAR
univarchar	SQL_WVARCHAR	SQL_C_CHAR
varbinary	SQL_VARBINARY	SQL_C_BINARY
Varchar	SQL_VARCHAR	SQL_C_CHAR

Advanced ASE features you can use with the ASE ODBC Driver:

- **Using Distributed Transactions:** ASE ODBC driver can be used to participate in two-phase commit transactions. It requires Microsoft Distributed Transaction Coordinator (MS DTC) be the transaction coordinator managing two-phase commit.
- **Using directory services:** Directory services allow the ASE ODBC Driver to get connection and other information from a central LDAP server; then, it uses this information to connect to an ASE server. It uses a property called Directory Service URL (DSURL) that indicates which LDAP server to use.
- **Using password encryption:** By default, the ASE ODBC Driver sends plain text passwords over the network to ASE for authentication. This default feature can be to change to encrypt passwords before they are sent over the network. When EncryptPassword = 1, the password is not sent over the wire until a login is negotiated; then, the password is encrypted and sent.
- **Using SSL:** When a client application requests a connection, the SSL-enabled server presents its certificate to prove its identity before data is transmitted.
- **Using failover in high availability systems:** A high availability cluster includes two or more machines that are configured so that if one machine (or application) is interrupted, the second machine assumes the workload of both machines. Each of these machines is called one node of the high availability cluster. A high availability cluster is used in an environment that must always be available.
- **Enabling Kerberos authentication:** Kerberos is an industry standard network authentication system that provides simple login authentication as well as mutual login authentication. It is used for single sign-on across various applications in extremely secure environments. Instead of passing passwords around the network, a Kerberos server holds encrypted versions of the passwords for users as well as available services. Adaptive Server and the ASE ODBC driver provide support for Kerberos connections.

Following is a list of connection parameters other than from the DSN parameter that can be supplied to the ASE ODBC Driver:

Property names	Description	Required	Default value
UID, UserID	A case-sensitive user ID required to connect to the ASE server.	Yes	Empty
PWD, Password	A case-sensitive password to connect to the ASE server.	No, if the user name does not require a password	Empty
Server	The name or IP address of the ASE server.	Yes	Empty
Port	The port number of ASE server.	Yes	Empty
AnsiNull	Strict ODBC compliance where you cannot use "= NULL." Instead, you must use "IsNull."	No	1
ApplicationName	The name used by ASE to identify the client application.	No	Empty
BufferCacheSize	Keeps the input / output buffers in pool. When large results will occur, increase this value to boost performance.	No	20
CharSet	The designated character set. The specified character set must be installed on the ASE server. Starting with version 15.0, the charset behavior in ODBC drivers for Linux has changed. If you want to use the ASE server's charset, you must specify it in the connection properties with **Charset=ServerDefault**.	No	Empty
ClientHostName	The name of the client host passed in the login record to the server.	No	Empty
ClientHostProc	The identity of client process on this host machine passed in the login record to the server.	No	Empty
CRC	By default, the driver returns the total records updated when multiple update statements are executed in a stored procedure. This count will also include all updates happening as part of the triggers set on an update or an insert. Set this property to 0 if you want the driver to return only the last update count.	No	1

Property names	Description Required	Default	value
Database	The database to which you want to connect.	No	Empty
DataIntegrity	Enables Kerberos Data Integrity.	No	0 (disabled)
DSPassword	The password used to authenticate on the LDAP server, if the LDAP server does not allow anonymous access. The password can be specified in the Directory Service URL (DSURL) as well.	No	Empty
DSPrincipal	The user name used to authenticate on the LDAP server, if the LDAP server does not allow anonymous access. The principal can be specified in the DSURL as well.	No	Empty
DSURL	The URL to the LDAP server.	No	Empty
DTCProtocol (Windows® only)	Allows the driver to use either an XA protocol or OleNative protocol when using distributed transactions.	No	XA
DynamicPrepare	When set to 1, the driver sends SQLPrepare calls to ASE to compile/prepare. This can boost performance if you use the same query repeatedly.	No	0
EnableServerPacketSize	Allows ASE server versions 15.0 or later to choose the optimal packet size.	No	1
EncryptedPassword	Specifies if password encryption is enabled: 0 indicates password encryption is disabled, 1 indicates password encryption is enabled. No	0	Encryption
The designated encryption. Possible values: ssl.	No	Empty	FetchArraySize
Specifies the number of rows the driver retrieves when fetching results	from the server.	No	25
HASession	Specifies if high availability is enabled. 0 indicates high availability disabled, 1 high availability enabled.	No	0

Property names	Description	Required	Default value
IgnoreErrorsIfRS Pending	Specifies whether the driver is to continue processing or stop if error messages are present. When set to 1, the driver will ignore errors & continue processing the results if more results are available from the server. When set to 0, the driver will stop processing the results if an error is encountered even if there are results pending.	No	0
UseCursor	Specifies whether cursors are to be used by the driver. 0 indicates do not use cursors, and 1 indicates use cursors.	No	0
Language	The language in which ASE returns error messages.	No	Empty – ASE uses English by default
LoginTimeOut	Number of seconds to wait for a login attempt before returning to the application. If set to 0, the timeout is disabled, and a connection attempt waits for an indefinite period of time.	No	10
MutualAuthentication	Enables Kerberos Mutual Authentication.	No	0 (disabled)
PacketSize	The number of bytes per network packet transferred between ASE and the client.	No	Server determined when driver is connected to ASE 15.0 or later. For older ASE servers, the default is 512.
QuotedIdentifier	Specifies if ASE treats character strings enclosed in double quotes as identifiers: 0 = do not enable quoted identifiers 1 = enable quoted identifiers	No	0
ReplayDetection	Enables Kerberos Replay Detection.	No	0
RestrictMaximum PacketSize	If you have memory constraints when **EnableServerPacketSize** is set to 1, then set this property to an **int** value in multiples of 512 to a maximum of 65536.	No	0
SecondaryPort	The port number of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1.	Empty

Property names	Description	Required	Default value
SecondaryServer	The name or the IP address of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1.	Empty
ServerInitiated Transactions	When SQL_ATTR_AUTOCOMMIT is set to "1" Adaptive Server starts managing transactions as needed. The driver issues a "set chained on" command on the connection. Older ODBC Drivers do not use this feature and manage the job of starting transactions by calling begin tran. Set this property to "0" if you want to maintain the old behavior or require that your connection not use "chained" transaction mode.	No	1
TextSize	The maximum size of binary or text data that will be sent over the wire.	No	Empty – ASE default is 32K.
TightlyCoupled Transaction (Windows only)	When using distributed transactions, if you are using two DSNs which connect to the same ASE server, set this to 1.	No	0
TrustedFile	If encryption is set to ssl, this property should be set to the path to the Trusted File.	No	Empty

7.2 ADO.NET Client Application

ASE ADO.NET Data Provider is an ADO.NET provider for the Sybase Adaptive Server Enterprise (ASE) database. It allows you to access data in ASE using any language supported by .NET, such as C#, Visual Basic .NET, C++ with managed extension, and J#. For more detail of ASE ADO.NET Data Provider, see "ASE ADO.NET Data Provider API Interface".

Connection String parameters:

Property names	Description	Required	Default value
UID, UserID, User ID, User	A case-sensitive user ID required to connect to the ASE server.	Yes	Empty
PWD, Password	A case-sensitive password to connect to the ASE server.	No, if the user name does not require a password	Empty
Server, Data Source, DataSource, Address, Addr, Network Address, Server Name	The name or the IP address of the ASE server.	Yes	Empty
Port, Server Port	The port number of ASE server.	Yes, unless the port number is specified in the Datasource	Empty
AnsiNull	Strict ODBC compliance where you cannot use "= NULL". Instead you have to use "IsNull". Set to 1 if you want to change the default behavior.	No	0
ApplicationName, Application Name	The name to be used by ASE to identify the client application	No	Empty
BufferCacheSize	Keeps the Input / Output buffers in pool. Increase for very large results to boost performance	No	20
ClientHostName	The name of client host passed in the login record to the server, for example: `ClientHostName='MANGO'`	No	Empty
ClientHostProc	The identity of client process on this host machine passed in the login record to the server, for example: `ClientHostProc='MANGO-PROC'`	No	Empty
CumulativeRecordCount, Cumulative Record Count, CRC	By default the driver (use provider for ADO.NET) returns the total records updated when multiple update statements are executed in a stored procedure. This count includes all updates happening as part of the triggers set on an update or an insert. Set this property to 0 if you want the driver to return only the last update count.	No	1

Property names	Description	Required	Default value
Database, Initial Catalog	The database to which you want to connect.	No	Empty
DSPassword, Directory Service Password	The password used to authenticate on the LDAP server, if the LDAP server does not allow anonymous access. The password can be specified in the DSURL as well.	No	Empty
DSPrincipal, Directory Service Principal	The user name used to authenticate on the LDAP server, if the LDAP server does not allow anonymous access. The Principal can be specified in the DSURL as well.	No	Empty
DSURL, Directory Service URL	The URL to the LDAP server.	No	Empty
DTCProtocol (Windows only)	Allows the driver to use either an XA protocol or OleNative protocol when using distributed transactions.	No	XA
EnableServerPacketSize	Allows ASE server versions 15.0 or later to choose the optimal packetsize.	No	1
EncryptPassword, EncryptPwd, Encrypt Password	Specifies if password encryption is enabled: 0 indicates password encryption is disabled, 1 indicates password encryption is enabled.	No	0
Encryption	The designated encryption. Possible values: **ssl, none**.	No	Empty
FetchArraySize	Specifies the number of rows the driver retrieves when fetching results from the server.	No	25
HASession	Specifies if High Availability is enabled. 0 indicates High Availability disabled, 1 High Availability enabled.	No	0
IgnoreErrorsIfRS Pending	Specifies whether the driver is to continue processing or stop if error messages are present. When set to 1, the driver will ignore errors & continue processing the results if more results are available from the server. When set to 0, the driver will stop processing the results if an error is encountered even if there are results pending.	No	0

Property names	Description	Required	Default value
Language	The Language in which ASE returns error messages.	No	Empty – ASE uses English by default.
LoginTimeOut, Connect Timeout, Connection Timeout	Number of seconds to wait for a login attempt before returning to the application. If set to 0 the timeout is disabled and a connection attempt waits for an indefinite period of time.	No	10
min pool size	You can force the provider to close connections to ASE so that total number of open connections hover around min pool size. The provider closes the connection on AseConnection.Close () if the number of connections in the pool are equal to min pool size.	No	20
max pool size	You can restrict the connection pool not to grow more than the specified max pool size. The provider will throw an AseException on AseConnection.Open() if this limit is reached.	No	100
NamedParameters	Set to false if you intend to use parameter markers instead of named parameters. Example with named parameters the SQL will look like SELECT * from titles where title_id = @title_id. The same SQL with parameter markers will look like SELECT * from titles where title_id =?	No	true
PacketSize, Packet Size	The number of bytes per network packet transferred between ASE and the client.	No	512
Ping Server	Set to false if you do not want the provider to verify that the connection is valid before it uses it from the connection pool.	No	true
Pooling	To disable connection pooling set to false.	No	true
QuotedIdentifier	Specifies if ASE treats character strings enclosed in double quotes as identifiers: 0 indicates do not enable quoted identifiers, 1 indicates enable quoted identifiers.	No	0

Property names	Description	Required	Default value
RestrictMaximum	PacketSizeIf you have memory constraints when EnableServerPacketSize is set to 1, then set this property to an int value in multiples of 512 to a maximum of 65536.	No	0
SecondaryPort, Secondary Port, Secondary Server Port	The port number of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1, unless the port number is specified in the Secondary DataSource.	Empty
SecondaryServer, Secondary Data Source, Secondary DataSource, Secondary Server, Secondary Address, Secondary Addr, Secondary Network Address, Secondary Server Name, Secondary Service Name	The name or the IP address of the ASE server acting as a failover server in an active-active or active-passive setup."Secondary Data Source" can be: SecondaryServer:SecondaryPort or SecondaryServer, SecondaryPort.	Yes, if HASession is set to 1.	Empty
TextSize	The maximum size of binary or text data in bytes that will be sent to or received from ASE, for example: TextSize=64000 sets this limit to 64K bytes.	No	Empty. ASE default is 32K.
TightlyCoupled Transaction (Windows only)	When using distributed transactions, if you are using two DSNs which connect to the same ASE server, set this to 1.	No	0
TrustedFile	If Encryption is set to ssl, this property should be set to the path to the Trusted File.	No	Empty
UseCursor, Use Cursor	Specifies whether cursors are to be used by the driver. 0 indicates do not use cursors and 1indicates use cursors.	No	0

7.3 OLE DB Client Application

Applications based on OLE DB are also able to connect to the most of the popular databases. Thus, application conversion is very easy. To convert an MySQL OLE DB client application to a corresponding Sybase OLE DB client application, the developer needs to change the ODBC client application, and for that follow the steps:

- Install the correct OLE DB for Sybase
- Check the version of the OLE DB client application and determine if all OLE D statements are supported.
- When an application connects to a database, it uses a set of connection parameters to define the connection. Connection parameters include information such as the server name, the database name, and a user ID. Change these connection parameters to connect to Adaptive Server with correct database.
- Possible changes in calling stored procedures
- Possible logical changes required to retain the program flow

There are two OLE DB providers you can use to access ASE:

- **Sybase ASE OLE DB Provider** The ASE OLE DB Provider provides access to ASE as an OLE DB data source without the need for ODBC components. The short name for this provider is **ASEOLEDB**.
- **Microsoft OLE DB provider for ODBC** Microsoft provides an OLE DB provider with a short name of **MSDASQL**. The **MSDASQL** provider makes ODBC data sources appear as OLE DB data sources. To do this, it requires the ASE ODBC Driver.

ASE OLE DB Provider datatype mappings:

ASE datatype	OLE DB datatype	C++ datatype
binary	DBTYPE_BYTES	unsigned char[]
bigint	DBTYPE_I8	long long
bit	DBTYPE_BOOL	BOOL
char	DBTYPE_STR, DBTYPE_BSTR	char[], BSTR
date	DBTYPE_DBDATE	DATE_STRUCT
datetime	DBTYPE_DBTIMESTAMP	TIMESTAMP_STRUCT
decimal	DBTYPE_DECIMAL	SQL_NUMERIC
double	DBTYPE_R8	double
float (<16)	DBTYPE_R4	float
image	DBTYPE_IUNKNOWN, DBTYPE_BYTES	IUnknown, unsigned char [] Sybase recommends that you use streams through IUnknown interfaces. It can also be bound as unsigned char [].
int [eger]	DBTYPE_I4	long
money	DBTYPE_CY	long long
nchar	DBTYPE_STR, DBTYPE_BSTR	char [], BSTR
numeric	DBTYPE_NUMERIC	SQL_NUMERIC
nvarchar	DBTYPE_STR, DBTYPE_BSTR	char [], BSTR
real	DBTYPE_R4	float
smalldatetime	DBTYPE_DBTIMESTAMP	TIMESTAMP_STRUCT

ASE datatype	OLE DB datatype	C++ datatype
smallint	DBTYPE_I2	short
smallmoney	DBTYPE_CY	long long
text	DBTYPE_IUNKNOWN, DBTYPE_STR IUnknown, char []	Sybase recommends that you use streams through IUnknown interfaces. It can also be bound as char [].
time	DBTYPE_DBTIME	TIME_STRUCT
timestamp	DBTYPE_BYTES	unsigned char []
tinyint	DBTYPE_UI1	unsigned char
unichar	DBTYPE_WSTR, DBTYPE_BSTR	wchar_t [], BSTR
unitext	DBTYPE_IUNKNOWN, DBTYPE_WSTR	IUnknown, wchar_t [] Sybase recommends that you use IUnknown.
univarchar	DBTYPE_WSTR, DBTYPE_BSTR	wchar_t [], BSTR
unsignedbigint	DBTYPE_UI8	unsigned long long
unsignedint	DBTYPE_UI4	unsigned long
unsignedsmallint	DBTYPE_UI2	unsigned short
varbinary	DBTYPE_BYTES	unsigned char[]
varchar	DBTYPE_STR, DBTYPE_BSTR	char [], BSTR

Advanced ASE features you can use with the ASE OLE DB Provider

- **Directory Services:** Using directory services, the ASE OLE DB Provider can get connection and other information from a central LDAP server to connect to an ASE server. It uses a property called **Directory Service URL (DSURL)** that indicates which LDAP server to use.
- **Password Encryption** By default, the ASE OLE DB Provider sends plain text passwords over the network to ASE for authentication. You can use this feature to change the default and encrypt passwords before they are sent over the network. When **EncryptPassword** is set to 1, the password is not sent over the wire until a login is negotiated; then, the password is encrypted and sent.
- **Data Encryption using SSL** When a client application requests a connection, the SSL-enabled server presents its certificate to prove its identity before data is transmitted
- **Kerberos Authentication** Kerberos is an industry standard network authentication system that provides simple login authentication as well as mutual login authentication. Kerberos provides user and service authentication. Kerberos is used for single sign-on across various applications in extremely secure environments. Instead of passing passwords around the network, a Kerberos server holds encrypted versions of the passwords for users and available services. Adaptive Server and the ASE OLE DB provider provide support for Kerberos connections. The ASE OLE DB provider specifically supports MIT, CyberSafe, and Active Directory KDCs.

List of connection parameters that can be supplied to the ASE OLE DB Provider:

Property names	Description	Required	Default value
User ID, UserID, UID	A case-sensitive user ID required to connect to the ASE server.	Yes	None
PWD, Password	A case-sensitive password to connect to the ASE server.	No, if the user name does not require a password.	Empty
Server	The name or the IP address of the ASE server.	No, if data source is specified.	Empty
Port	The port number of the ASE server.	No, if data source is specified.	Empty
AnsiNull	Strict compliance where you cannot use "= NULL." Instead, you must use "IsNull."	No	1
ApplicationName	The name ASE uses to identify the client application.	No	Empty
BufferCacheSize	Keeps the input and output buffers in pool. When large results will occur, increase this value to boost performance.	No	20
CharSet	The designated character set. The specified character set must be installed on the ASE server. Starting with version 15.0, the charset behavior has changed. If you want to use the ASE server's charset, you must specify it in the connection properties with Charset=ServerDefault. Otherwise, the driver picks up the charset, which the client specifies, or gets it from the system environment variables in the order of LC_CTYPE and LANG. If both the environment variables have not been defined, it uses the value iso_1.	No	Empty
ClientHostName	The name of the client host passed in the login record to the server.	No	Empty
ClientHostProc	The identity of the client process on this host machine passed in the login record to the server.	No	Empty

Property names	Description	Required	Default value
CRC	By default, the driver returns the total records updated when multiple update statements are executed in a stored procedure. This count will also include all updates happening as part of the triggers set on an update or an insert. Set this property to 0 if you want the driver to return only the last update count.	No	1
DataIntegrity	Enables Kerberos Data Integrity.	No	0 (disabled)
Data Source	The Data Source you want to connect in *Server:Port* format.	No, if server and port are specified.	Empty
DSPassword	The password used to authenticate on the LDAP server, if the LDAP server does not allow anonymous access. The password can be specified in the DSURL as well.	No	Empty
DSPrincipal	The user name used to authenticate on the LDAP server, if the LDAP server does not allow anonymous access. The Principal can be specified in the DSURL as well.	No	Empty
DSURL	The URL to the LDAP server.	No	Empty
DynamicPrepare	When set to 1, the driver sends **SQLPrepare** calls to ASE to compile/prepare. This can boost performance if you use the same query repeatedly.	No	0
EnableServerPacketSize	Allows ASE server versions 15.0 or later to choose the optimal packetsize.	No	1
EncryptedPassword	Specifies if password encryption is enabled: 0 indicates password encryption is disabled, 1 indicates password encryption is enabled.	No	0
Encryption	The designated encryption. Possible values: **ssl**.	No	Empty
HASession	Specifies if high availability is enabled: 0 indicates high availability disabled, 1 high availability enabled.	No	0

Property names	Description	Required	Default value
Initial Catalog, Database	The database to which you want to connect.	No	Empty
Language	The language in which ASE returns error messages.	No	Empty – ASE uses English by default.
LoginTimeOut	Number of seconds to wait for a login attempt before returning to the application. If set to 0, the timeout is disabled and a connection attempt waits for an indefinite period of time.	No	10
MutualAuthentication	Enables Kerberos Mutual Authentication.	No	0 (disabled)
PacketSize	The number of bytes per network packet transferred between ASE and the client.	No	Server determined when driver is connected to ASE 15.0 or later. For older ASE servers the default is 512.
QuotedIdentifier	Specifies if ASE treats character strings enclosed in double quotes as identifiers: 0 indicates do not enable quoted identifiers, 1 indicates enable quoted identifiers.	No	0
ReplayDetection	Enables Kerberos Replay Detection.	No	0
RestrictMaximum PacketSize	If you have memory constraints when EnableServerPacketSize is set to 1, then set this property to an int value in multiples of 512 to a maximum of 65536.	No	0
SecondaryPort	The port number of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1	Empty
SecondaryServer	The name or the IP address of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1	Empty

Property names	Description	Required	Default value
ServerInitiated Transactions	When SQL_ATTR_AUTOCOMMIT is set to "1," Adaptive Server starts managing transactions as needed. The driver issues a **set chained on** command on the connection. Older ODBC Drivers do not use this feature and manage the job of starting transactions. Set this property to''0' if you want to maintain the old behavior or require that your connection not use "chained" transaction mode.	No	1
TextSize	The maximum size of binary or text data that will be sent over the wire.	No	Empty. ASE default is 32K.
TrustedFile	If encryption is set to ssl, this property should be set to the path to the Trusted File.	No	Empty
UseCursor	Specifies whether cursors are to be used by the driver: 0 indicates do not use cursors, and 1 indicate use cursors.	No	0

7.4 JDBC Client Application

In this scenario, the developer needs to change the JDBC client application, and for that follow the steps:

- Install the correct JDBC driver for Sybase.
- To convert a MySQL JDBC client application to a corresponding Sybase JDBC client application, the developer needs to check the version of the JDBC client application and determine if all JDBC statements are supported.
- Determine how application connects to the data source. Connections can be achieved using either DataManager class or with a datasource implementation.
- There are two ways to set the driver connection properties:
  a. Use the **DriverManager.getConnection** method in your application.
  b. Set the connection properties when you define the URL.
  Example of Sybase JDBC Connection:

```
import com.sybase.jdbcx.SybDriver;

SybDriver sybDriver = (SybDriver)
  Class.forName   ("com.sybase.jdbc3.jdbc.SybDriver"). newInstance();

sybDriver.setVersion (com.sybase.jdbcx.SybDriver.VERSION_6);

DriverManager.registerDriver (sybDriver);

//connect to database
Properties props = new Properties ();
```

```
 props. put  ("user", "userid");
 props .put ("password", "user_password");

/*
* Make sure you set connection properties before
* attempting to make a connection. You can also
* set the properties in the URL.
*/
Connection con = DriverManager.getConnection
  ("jdbc: sybase:Tds:host:port", props);
```

- If the connectivity is obtained through a datasource, the application server should be properly configured.
- If the SQL statements passed to the JDBC methods use any MySQL-specific syntax, then these items will need to be updated to reflect T-SQL syntax of Sybase.

List of connection properties for jconnect that can be supplied to the ASE

Property	Description	Default value
APPLICATIONNAME	Specifies an application name. This is a user-defined property. The server side can be programmed to interpret the value given to this property.	Null
BE_AS_JDBC_COMPLIANT_ AS_POSSIBLE	Adjusts other properties to ensure that jConnect methods respond in a way that is as compliant as possible with the JDBC 3.0 standard. These properties are affected (and overridden) when this property is set to "true": CANCEL_ALL (set to "false") LANGUAGE CURSOR (set to "false") SELECT_OPENS_CURSOR (set to "true") FAKE_METADATA (set to "true") GET_BY_NAME_USES_COLUMN_LABEL (set to "false")	False
CACHE_COLUMN_METADATA	To improve performance when you are setting DYNAMIC_PREPARE to "true," jConnect will cache **ResultSet Metadata** on consecutive executions.	None
CANCEL_ALL	Determines the behavior of the **Statement.cancel** method..	Depends on version setting.

Property	Description	Default value
CAPABILITY_TIME	Used only when JCONNECT_VERSION >= 6. When jConnect is connected to a server that supports the *TIME* datatype, and all parameters of type *java.sql.Time* or *escape literals {t ...}* are processed as TIME. Previous versions of jConnect treat such parameters as *DATETIME* and prepend '1970-01-01' to the *java.sql.Time* parameter. If the underlying datatype is *datetime* or *smalldatetime* the date part also gets stored in the database. In the new versions of jConnect when *TIME* is processed, then the server converts time to the underlying datatype and will prepend its own base year. This can result in incompatibilities between old and new data. If you are using *datetime* or *smalldatetime* datatypes for *java.sql.Time*, then for backward compatibility you should leave **CAPABILITY_TIME** as *false*. Leaving this property as false forces jConnect to process *java.sql.Time* parameters or *escape literals {t...}* as *DATETIME* regardless of the server capability of handling *TIME* datatype. Setting this property to *true* will cause jConnect to process *java.sql.Time* parameters as *TIME* datatype when connected to ASE 12.5.1 or later. Sybase recommends you leave this property as *false* if you are using *smalldatetime* or *datetime* columns to store time values.	False
CAPABILITY_WIDETABLE	If you do not require JDBC ResultSetMetaData like Column name as a performance improvement, you can set this to “false.” The result is that less data is exchanged over the network and increases performance. Unless you are using EAServer, Sybase recommends, that you use the default setting.	False
CHARSET	Specifies the character set for strings passed to the database. If the CHARSET value is **null**, jConnect uses the default character set of the server to send *string* data to the server. If you specify a CHARSET, the database must be able to handle characters in that format. If the database cannot do so, a message is generated indicating that character conversion cannot be properly completed. When using jConnect 6.05 with DISABLE_UNICHAR_SENDING set to “false”, jConnect detects when a client is trying to send characters to the server that cannot be represented in the character set that is being used for the connection. When that occurs, jConnect sends the character data to the server as *unichar* data, which allows clients to insert Unicode data into *unichar/univarchar* columns and parameters.	Null
CHARSET_CONVERTER_CLASS	Specifies the character-set converter class you want jConnect to use. jConnect uses the version setting from **SybDriver.setVersion**, or the version passed in with the JCONNECT_VERSION property, to determine the default character-set converter class to use.	Version-dependent

Property	Description	Default value
CLASS_LOADER	A property you set to a DynamicClassLoader object that you create. The DynamicClassLoader is used to load Java classes that are stored in the database but which are not in the CLASSPATH at application start-up time.	Null
CONNECTION_FAILOVER	Used with the Java Naming and Directory Interface (JNDI).	True
DISABLE_UNICHAR_SENDING	When a client application sends *unichar* characters to the server (along with non-*unichar* characters), there is a slight performance hit for any character data sent to the database. This property defaults to "false" in jConnect 6.05. Clients using older versions of jConnect who wish to send *unichar* data to the database must set this property to "false".	Version-dependent
DISABLE_UNPROCESSED_ PARAM_WARNINGS	Disables warnings. During results processing for a stored procedure, jConnect often reads return values other than row data. If you do not process the return value, jConnect raises a warning. To disable these warnings (which can help performance), set this property to "true."	False
DYNAMIC_PREPARE	Determines whether dynamic SQL prepared statements are precompiled in the database.	False
ENABLE_SERVER_PACKETSIZE	Allows you to specify if you want to use server specified packet size. By default this property is set to "true" that uses the server specified packet size.	True
ENCRYPT_PASSWORD	Allows a secure login. When it is set to "true," both login and remote site passwords are encrypted and sent to the server.	False
ESCAPE_PROCESSING_DEFAULT	Circumvents processing of JDBC function escapes in SQL statements. By default, jConnect parses all SQL statements submitted to the database for valid JDBC function escapes. If your application is not going to use JDBC function escapes in its SQL calls, you can set this connection property to "false" to avoid this processing. This can provide a slight performance benefit.	True
EXPIRESTRING	Contains the license expiration *date*. Expiration is never except for evaluation copies of jConnect. This is a read-only property.	Never
FAKE_METADATA	Returns phony metadata. When you call the **ResultSetMetaData** methods **getCatalogName**, **getSchemaName**, and **getTableName** and this property is set to "true," the call returns empty strings ("") because the server does not supply useful metadata. When this property is set to "false," calling these methods throws a "Not Implemented" SQLException. If you have enabled wide tables and are using an Adaptive Server 12.5 or later, this property setting is ignored, because the server does supply useful metadata.	False

Property	Description	Default value
GET_BY_NAME_USES_ COLUMN_LABEL	Provides backward compatibility with versions of jConnect earlier than 6.0. With Adaptive Server version 12.5, jConnect has access to more metadata than was previously available. Previous to version 12.5, *column name* and *column alias* meant the same thing. jConnect can now differentiate between the two when used with a 12.5 or later Adaptive Server with wide tables enabled. To preserve backward compatibility, set this property to "true."If you want calls to getByte, getInt, get* (*String columnName*) to look at the actual name for the column (called for in the JDBC 2.0 specification), set this property to "false."	True
GSSMANAGER_CLASS	Specifies a third-party implementation of the org.ietf.jgss.GSSManager class. This property can be set to a string or a GSSManager object. If the property is set to a string, the value should be the fully qualified class name of the third-party GSSManager implementation. If the property is set to an object, the object must extend the org.ietf.jgss.GSSManager class.	Null
HOSTNAME	Identifies the name of the current host.	None. The max length is 30 characters and, if exceeded, it is truncated to 30.
HOSTPROC	Identifies the application process on the host machine.	None
IGNORE_DONE_IN_PROC	Determines that intermediate update results (as in stored procedures) are not returned, only the final result set.	False
IS_CLOSED_TEST	Allows you to specify what query, if any, is sent to the database when **Connection.isClosed** is called. For additional information,	Null
JCONNECT_VERSION	Sets version-specific characteristics.	6
LANGUAGE	Sets the language for error messages returned from the server and for jConnect messages. The setting must match a language in *syslanguages*.	
LANGUAGE_CURSOR	Determines that jConnect uses “language cursors” instead of “protocol cursors.”	False
LITERAL_PARAMS	When set to "true," any parameters set by the **setXXX** methods in the **PreparedStatement** interface are inserted literally into the SQL statement when it is executed. If set to "false," parameter markers are left in the SQL statement and the parameter values are sent to the server separately.	False

Property	Description	Default value
USE_METADATA	Creates and initializes a **DatabaseMetaData** object when you establish a connection. The **DatabaseMetaData** object is necessary to connect to a specified database. jConnect uses **DatabaseMetaData** for some features, including Distributed Transaction Management support (JTA/JTS) and dynamic class loading (DCL). If you receive error 010SJ, which indicates that your application requires metadata, install the stored procedures for returning metadata that come with jConnect.	True
PACKETSIZE	Identifies the network packet size. If you are using ASE 15.0, Sybase recommends that you do not set this property and allow jConnect and ASE 15.0 to pick the network packet size that is appropriate for your environment.	512
PASSWORD	Identifies the login password. Set automatically if using the **getConnection** (**String, String, String**) method, or explicitly if using **getConnection** (**String, Props**).	None
PRELOAD_JARS	Contains a comma-separated list of *.jar* file names that are associated with the CLASS_LOADER that you specify. These *.jar* files are loaded at connect time, and are available for use by any other connection using the same jConnect driver.	Null
PROTOCOL_CAPTURE	Specifies a file for capturing TDS communication between an application and an Adaptive Server.	Null
PROXY	Specifies a gateway address. For the HTTP protocol, the URL is http://host:port. To use the HTTPS protocol that supports encryption, the URL is https://host:port/servlet_alias.	None
QUERY_TIMEOUT_CANCELS_ALL	Forces jConnect to cancel all Statements on a Connection when a read timeout is encountered. This behavior can be used when a client has calls **execute ()** and the timeout occurs because of a deadlock (for example, trying to read from a table that is currently being updated in another transaction). The default value is false. Depending on future discussions with Sun, this property may be lumped in with the property values affected by the BE_AS_JDBC_COMPLIANT_AS_POSSIBLE property.	False
REMOTEPWD	Contains remote server passwords for access through server-to-server remote procedure calls.	None
REPEAT_READ	Determines whether the driver keeps copies of columns and output parameters so that columns can be read out of order or repeatedly.	True

Property	Description	Default value
REQUEST_HA_SESSION	Indicates whether the connecting client wants to begin an high availability (HA) failover session with a version 12 or later Adaptive Server configured for failover. Setting this property to "true" causes jConnect to attempt a failover login. If you do not set this connection property, a failover session does not start, even if the server is configured for failover. You cannot reset the property once a connection has been made. If you want more flexibility for requesting failover sessions, code the client application to set REQUEST_HA_SESSION at runtime.	False
REQUEST_KERBEROS_SESSION	Determines whether jConnect uses Kerberos for authentication. If this property is set to "true," a value for the SERVICE_PRINCIPAL_NAME property must also be specified. You may also wish to provide a value for the GSSMANAGER_CLASS property.	False
RMNAME	Sets the Resource Manager name when using distributed transactions (XA). This property overrides a Resource Manager name that may be set in an LDAP server entry.	Null
SECONDARY_SERVER_ HOSTPORT	Sets the hostname and port for the secondary server when the client is using an HA failover session. The value for this property should be in the form of **hostName: portNumber**. This property is ignored unless you have also set REQUEST_HA_SESSION to "true	Null
SELECT_OPENS_CURSOR	Determines whether calls to **Statement.executeQuery** automatically generate a cursor when the query contains a **FOR UPDATE** clause. If you have previously called **Statement.setFetchSize** or **Statement.setCursorName** on the same statement, a setting of "true" for SELECT_OPENS_CURSOR has no effect. You may experience some performance degradation when SELECT_OPENS_CURSOR is set to "true."	False
SERIALIZE_REQUESTS	Determines whether jConnect waits for responses from the server before sending additional requests.	False
SERVER_INITIATED_ TRANSACTIONS	Allows the server to control transactions. By default the property is set to “true” and jConnect lets the server start and control transactions by using **Transact-SQL™** set **chained on**.” If set to “false,” the transactions are started and controlled by jConnect by using transact sql "begin tran." Sybase recommends that you allow the server to control the transactions.	True
SERVICENAME	Indicates the name of a back-end database server that a DirectConnect gateway serves. Also used to indicate which database should be used upon connecting to Adaptive Server Enterprise.	None

Property	Description	Default value
SERVERTYPE	When connected to OpenSwitch set this property to "OSW." This allows jConnect to send certain instructions to OpenSwitch that allows OpenSwitch to remember initial connection settings for example, isolation level, textsize, quoted identifier and autocommit when OpenSwitch is moved to a different server instance.	None
SERVICE_PRINCIPAL_NAME	Used when establishing a Kerberos connection to Adaptive Server Enterprise. The value of this property should correspond both to the server entry in your Key Distribution Center (KDC) and to the server name under which your database is running. The value of the SERVICE_PRINCIPAL_NAME property is ignored if the REQUEST_KERBEROS_SESSION property is set to "false."	Null
SESSION_ID	A TDS session ID. When this property is set, jConnect assumes that an application is trying to resume communication on an existing TDS session held open by the TDS-tunneling gateway. jConnect skips the login negotiations and forwards all requests from the application to the specified session ID.	Null
SESSION_TIMEOUT	Specifies the amount of time (in seconds) that an HTTP-tunnelled session (created using the jConnect TDS-tunnelling servlet) remains alive while idle. After the specified time, the connection is automatically closed.	Null
SQLINITSTRING	Defines a set of commands to be passed to the database server when a connection is opened. These must be SQL commands that can be executed using the **Statement.executeUpdate** method.	Null
STREAM_CACHE_SIZE	Specifies the maximum size used to cache statement response streams.	Null (unlimited cache size)
SYBSOCKET_FACTORY	Enables jConnect to use your custom socket implementation. Set SYBSOCKET_FACTORY either to: The name of a class that implements **com.sybase.jdbcx.SybSocketFactory**; or “DEFAULT,” which instantiates a new **java.net.Socket()** Use this property to make an SSL connection to your database.	Null
TEXTSIZE	Allows you to set the TEXTSIZE. By default ASE and ASA allow 32627 bytes to be read from an image or text column. If you have the jConnect mda tables installed, jConnect changes that value to 2GB. However setting this value when connected to OpenSwitch allows the connection to remember the setting when OpenSwitch is moved to a different server instance.	2GB

Property	Description	Default value
USER	Specifies the login ID. Set automatically if using the **getConnection** (**String, String, String**) method, or explicitly if using **getConnection** (**String, Props**).	None
VERSIONSTRING	Provides read-only version information for the JDBC driver.	jConnect driver version

8 VALIDATE THE DATABASE SERVER MIGRATION

The final step in the migration process is to validate the migration with a through testing process. It is very important to validate every aspect of the database, as everything except the core database design has been transformed. The migrated database should be verified to ensure that all data is accessible, and was imported without any failure or modification that could cause applications to function improperly. 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. Testing should cover the Adaptive Server installation, security, database objects, data, and performance.

With all of this complete, the application can be considered to have successfully migrated from MySQL to Sybase ASE.

The database migrated is validated at three different stages of the migration:

a) After the database is migrated with architecture, users, roles, database objects and data.
b) After the database migrated is validated, the entire application with the migrated data is validated. All modules of the system and any additional applications (WEB, supportive modules, Java programs, etc.) running against the target database instance should be verified to ensure that there are no problems with the new environment.
c)Once verified the application performs as expected, validate the performance, stress and scalability of the application.

Validation of architecture, users, roles, data objects and data is describes in this section.

8.1 Physical architecture of the database

Test the database device; database and database logs are created of correct size and at correct location. Validate the configuration of the database.

8.2 Security

Validate the logins, users, created correctly with the correct role assigned.

8.3 Logical architecture of the database

Validate the database objects like tables, indexes, stored procedures, view, triggers, rules, constraints and rules created correctly.

8.4 Data

Validate the number and type of objects in MySQL and Adaptive Server Database. Validate the conversion of data types of the columns and the constraints applied on the columns of the table. Check the referential integrity for each table. During the data migration, validate the accurate and complete transfer of data by taking the count of rows in each table of both the database.

Sometimes indexes, constraints, triggers are dropped or disabled during the data migration for performance reasons. These data are not checked for correctness. If database has proper constraints in the database then database itself checks the integrity of the data when constraints are added or enabled. In many application constraints are built into the application. In such cases, verify data integrity by identifying the rules in the application.

8.5 Functionality

Validate all the component of Adaptive Server - the stored procedure, triggers; rules etc. created correctly comprising SQL code operate in the same manner as the original MySQL objects. Execution of test cases against these objects should produce identical results.

8.6 Performance

The response time and throughput of the application using Adaptive Server should be same or better than the application with MySQL database.

APPENDIX A – PORTABILITY CHECK

This checklist provides the developer with the overview of issues that will be encountered during the migration.

Make current application architecture clear:

- Create an overview of the existing application including all products and displaying all in- and outgoing data and the methods used
- Describe the fallback procedures used
- Describe the security method used

Find Sybase compatible products and methods for all components:

- RDBMS
- API
- JDBC
- ODBC

Make plan to address each component:

- If the component needs to be replaced:
  - install component
  - create new configuration/code
- If it can be reused:
  - check for possible changes in configuration or code

Assign a lot of attention and time when converting the application:

1. Conversion of the application when business logic and database access code is mixed in a non-modular design.
2. Depending on complexity of the application, the data access code and business logic can be placed in database by creating stored procedure, triggers, rules, constraints etc. Using stored procedure, which encapsulates data access in reusable blocks of code that are precompiled and have their own permission sets; the database access can be centralized along with the business logic. It reduces the maintenance costs by placing all of the business logic in a central repository. Also, putting logic in the database ensures that every application that accesses the database will follow the same set of business rules.

APPENDIX B – MIGRATION CHECKLIST

After the architecture, data and application are deployed, every aspect of deployment must be validated to ensure the successful migration. The following checklist lists all actions needed to perform a successful migration.

Data Migration:

- Creation of Sybase Adaptive server instance, devices, database(s) and segments
- Check the setting of 'sa' password as by default it is set as NULL.
- Check for appropriate database options set
- Creation of security – logins, roles and permissions
- Creation the database objects
- Migration of data
- Running of update statistics
- Check the maximum users connection set appropriately
- Establish schedule for transaction log dumping that will be sufficient to recover the activity to the satisfaction of the business.
- Check the tuning of the database and memory usage settings applied
- Check the sufficient disk space for the expected size of the database 12 months ahead.
- Check the sufficient size of the tempdb database for handling temporary working storage, sort activity and for temporary user tables.
- Check sufficient size of transaction log of the user database created.
- Database monitoring activities
- Documentation of database and server configuration settings.

Application Migration:

- Recreate stored procedure and triggers
- Check all the SQL used in the application code for SQL language differences
- Check all the Java bases scripts for specific SQL
- Check for Business objectives and requirements met
- Verify the feed from other systems under load to ensure that system availability is unaffected at the time they run
- Check the security settings for the application set
- Check the required environment setting for the application set.
- Integration testing of the Application with third party system
- Acceptable Processing speed

Migrate all other components found in the architecture overview of Appendix A

Validate Migration:

- Use sp_sysmon and sp_objectstats to check for any system bottlenecks
- Count the number objects in both the database –MySQL and Sybase
- Count the number of rows of each table in both the database

Inform Users if any, change in the Front-end

APPENDIX C – DATA TYPE MAPPING

MySQL Datatypes:

Data Type	**Size**	**Description**
CHAR[Length]	Length bytes	A fixed-length field from 0 to 255 characters long.
VARCHAR(Length)	String length + 1 bytes	A fixed-length field from 0 to 255 characters long.
TINYTEXT	String length + 1 bytes	A string with a maximum length of 255 characters.
TEXT	String length + 2 bytes	A string with a maximum length of 65,535 characters.
MEDIUMTEXT	String length + 3 bytes	A string with a maximum length of 16,777,215 characters.
LONGTEXT	String length + 4 bytes	A string with a maximum length of 4,294,967,295 characters.
TINYINT[Length]	1 byte	Range of -128 to 127 or 0 to 255 unsigned.
SMALLINT[Length]	2 bytes	Range of -32,768 to 32,767 or 0 to 65535 unsigned.
MEDIUMINT[Length]	3 bytes	Range of -8,388,608 to 8,388,607 or 0 to 16,777,215 unsigned.
INT[Length]	4 bytes	Range of -2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 unsigned.
BIGINT[Length]	8 bytes	Range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 unsigned.
FLOAT	4 bytes	A small number with a floating decimal point.
DOUBLE[Length, Decimals]	8 bytes	A large number with a floating decimal point.
DECIMAL[Length, Decimals]	Length + 1 or Length + 2 bytes	A DOUBLE stored as a string, allowing for a fixed decimal point.
DATE	3 bytes	In the format of YYYY-MM-DD.
DATETIME	8 bytes	In the format of YYYY-MM-DD HH:MM:SS.
TIMESTAMP	4 bytes	In the format of YYYYMMDDHHMMSS; acceptable range ends in the year 2037.
TIME	3 bytes	In the format of HH:MM:SS
ENUM	1 or 2 bytes	Short for enumeration, which means that each column can have one of several possible values.
SET	1, 2, 3, 4, or 8 bytes	Like ENUM except that each column can have more than one of several possible values.

Sybase Data types:

Sybase Data Type	Description	Conversion
char(n) 1<=n<=255	Fixed-length character data If n>255, SQL Server nchar(n) should be	converted to Sybase text
nchar(n) 1<=n<=255	Fixed-length character data (Unicode, multibyte sets)	If n>255, SQL Server nchar(n) should be converted to Sybase text
varchar (n) 1<=n<=255	Variable-length character data	If n>255, SQL Server varchar(n) should be converted to Sybase text
Nvarchar (n) 1<=n<=255	Variable-length character data (Unicode, multibyte sets)	If n>255, SQL Server nvarchar(n) should be converted to Sybase text
-	Integer, 64-bit	SQL Server bigint should be converted to Sybase decimal(19,0)
int	Integer, 32-bit	Not required
smallint	Integer, 16-bit	Not required
tinyint	Integer, 8-bit	Not required
datetime	Date and time	Not required
smalldatetime	Date and time	Not required
money	Monetary data	Not required
smallmoney	Monetary data	Not required

decimal(p, s)
1<=p<= 38; 0<=s<=p;
	Fixed-point numeric data	Not required

numeric (p, s) 1<=p<=38; 0<=s<=p;	Fixed-point numeric data	Not required
float(n) 1<=n<=machine dependent	Floating point numeric data	Not required
double precision	Floating point numeric data, 64-bit	Sybase double precision should be converted to SQL server float (53)
real	Floating point numeric data, 32-bit	Not required
text	Variable-length character data, up to 2 Gb	Not required
-	Variable-length Unicode character data, up to 1 Gb	SQL Server ntext should be converted to Sybase text
binary(n) 1<=n<=255	Fixed-length binary data	If n>255, SQL Server binary(n) should be converted to Sybase image
varbinary(n) 1<=n<=255	Variable-length binary data	If n>255, SQL Server varbinary(n) should be converted to Sybase image
image	Variable-length binary data, up to 2 Gb	Not required
bit	Integer value 0 or 1	SQL Server bit can be NULL, while Sybase bit cannot be NULL Nullable SQL Server bit should be converted to Sybase tinyint or char(1)
-	Globally unique identifier (GUID)	SQL Server uniqueidentifier should be converted to Sybase char(36)
timestamp	Unique number that gets updated every time a row gets updated	Not required

APPENDIX D – FUNCTION MAPPING

Many of the functions in MySQL Server and Adaptive Server Enterprise have the same name. Most MySQL functions that have different names have an equivalent Sybase ASE version. MySQL contains a few built-in functions that do not exist in ASE. Most of these functions can be created in ASE as user-defined functions that perform the same activity. If you give these functions the same name in ASE database, you will not need to modify the existing client applications SQL statements. Here are some examples of how Adaptive Server Enterprise user-defined functions can supply the same functionality as their MySQL built-in counterparts:

```
CREATE FUNCTION FROM_UNIXTIME (IN fromdt bigint default 0,
IN fmt varchar (32) default 'Mmm dd, yyyy hh:mm:ss' )
RETURNS datetime
BEGIN
RETURN (dateformat (dateadd (second, fromdt, '1970/01/01 00:00:00'), fmt))
END;
CREATE FUNCTION SEC_TO_TIME (IN sec bigint default 0)
RETURNS time
BEGIN
Return (dateadd (second, sec, '1970/01/01 00:00:00'))
END;
```

The following sections detail many of the MySQL functions along with their ASE equivalents. The list is extensive, but not exhaustive, as the list of functions in both MySQL and ASE changes with each release.

Numeric Functions:

MySQL	**Sybase ASE**	**Comments**
X DIV y	FLOOR (x/y)	
LN (x)	LOG (x)	
LOG (x, y)	LOG (x)/LOG (y)	
LOG2 (x)	LOG (x)/ LOG (2)	
POW (x, y)	POWER (x, y)	
ATAN	ATAN2 (x, y)	
CEIL (x)	CEILING (x)	
ROUND (x)	ROUND (x, 0)	

Date and Time Functions:

MySQL	Sybase ASE	Comments
TIME (expr)	CAST (expr as TIME)	The CONVERT function could also be used.
TIMESTAMP (expr)	DATETIME (expr)	
DAYOFWEEK (expr)	DOW (expr)	
WEEKDAY (expr)	MOD (DOW (expr)-1, 7)	
DAYOFMONTH (expr)	DAY (expr)	
WEEKOFYEAR (expr)	DATEPART (week, expr)	
PERIODADD (expr, N)	DATEFORMAT (DATEADD (month, 2, expr ||'/01'), 'YYYYMM')	
PERIOD_DIFF (P1, P2)	DATEDIFF (month, P2||'/01', P1||'/01')	
ADDDATE (date, numdays)	DATEADD (day, numdays, date)	
SUBDATE (date, numdays)	DATEADD (day, -numdays, date)	
EXTRACT (type FROM date)	DATEPART (type, date)	The 'type' argument differs between Adaptive Server and MySQL and will have to be adjusted accordingly.
TO_DAYS (date)	DAYS (date) –58	Adaptive Server measures differences in dates from '0000/02/29' instead of '0000/01/01'.
MAKEDATE(year, dayofyear)	YMD (year, 0, dayofyear)	
UNIX_TIMESTAMP (date)	DATEDIFF (second, '1981/01/01', date)	

String Functions:

MySQL	Sybase ASE	Comments
IFNULL (a, b)	IFNULL (a, b, a) or ISNULL (a, b)	The Adaptive Server IFNULL function behaves slightly differently from the MySQL version.
IF (cond, a, b)	IF cond THEN a ELSE b END IF	Adaptive Server supports the IF statement in both procedural logic, as well as embedded, as an expression in a select list.
CONCAT (a, b, ...)	STRING (a, b, ...)	In MySQL, if any of a, b, ... is NULL, the return value is NULL, while in Adaptive Server Enterprise if any of a, b, ... is NULL, it is treated as an empty string for the purposes of concatenation.
CONCAT_WS(sep, str1, str2, ...)	STRING(str1, sep, str2, sep, ...)	See comment for Concat () function above.

MySQL	Sybase ASE	Comments
CONV (N, frombase, tobase)	INTOHEX (N), HEXTOINT (N)	Adaptive Server only provides functions that allow you to convert to and from hexadecimal. Other conversions would have to be manually implemented using a UDF.
HEX (arg)	INTOHEX (srg) if arg is numeric, HEXTOINT (arg) if arg is string	
CHAR (N...)	CHAR (N)	The Adaptive Server CHAR () function only supports one argument.
STRCMP (expr1, expr2)	COMPARE (expr1, expr2)	
LENGTH (str)	BYTE_LENGTH (str)	The Adaptive Server Enterprise LENGTH () function returns the number of characters in str, not necessarily the byte length.
OCTET_LENGTH (str)	BYTE_LENGTH (str)	
CHARACTER_- LENGTH (str)	CHAR_LENGTH (str)	
BIT_LENGTH (str)	BYTE_LENGTH (str) * 8	
LOCATE (substr, str [, pos])	LOCATE (str, substr [, pos])	The order of the arguments differs in Adaptive Server Enterprise.
POSITION (substr IN str) INSTR (str, substr)	LOCATE(str, substr) LOCATE(str, substr)	
SUBSTRING (str FROM pos [FOR len])	SUBSTRING (str, pos [,len])	
MID (str, pos, len)	SUBSTRING (str, pos, len)	
TRIM (str)	TRIM (str)	Adaptive Server Enterprise does not support the other forms of the MySQL TRIM function.
INSERT (str, pos, len, newstr)	STUFF (str, pos, len, newstr)	
ELT (N, str1, str2, ...)	ARGN (N, str1, str2, ...)	
CONVERT (expr USING trans_name)	CSCONVERT (expr, trans_name)	

APPENDIX E – SQL LANGUAGE DIFFERENCES

Most of the syntax features of MySQL are available in Adaptive Server Enterprise, but occasionally the syntax for accessing those features is different. The following section details many of these statements along with their Sybase ASE equivalents.

Operators:

MySQL has several operators used to compare two or more arbitrary expressions and evaluate Boolean expressions. The following is a list of those expressions, along with the Sybase ASE equivalent if applicable.

MySQL Operator	**Sybase ASE Operator**	**Comments**
! =	<>	
⇔	(expr1 = expr2 OR ((expr1 IS NULL) AND (expr2 IS NULL)))	The ⇔ operator represents equality, including NULL values (NULL=NULL is true).
ISNULL (expr)	IS NULL expr	
INTERVAL (N, N1, N2, ...)	None built in	A user-defined function could easily be used to achieve the same function. For example: if (N < N1) then 0 elseif (N < N2) then 1 elseif...
!	NOT	
&&	AND	
||	OR	
a XOR b	((a AND (NOT b)) OR ((NOT a) AND b))	The Adaptive Server expression is complex for large numbers of XOR arguments, so an alternative method is recommended (Dependant on the specific application scenario) to migrate these expressions.

Data Manipulation Language:

MySQL Statement	**Sybase ASE Statement**	**Comments**
INSERT ...	INSERT ...	Adaptive Server Enterprise also offers the options ERROR and SKIP for existing rows.
ON DUPLICATE KEY UPDATE	ON EXISTING UPDATE	
SELECT ... INTO OUTFILE	UNLOAD SELECT ... DBISQL OUTPUT TO	
SELECT/UPDATE/DELETE ... LIMIT	FIRST or TOP n	
DEFAULT ‘0’ NOT NULL auto_increment	NOT NULL DEFAULT AUTOINCREMENT	
NOT NULL auto_increment	NOT NULL DEFAULT AUTOINCREMENT	
LIMIT offset, numRows	TOP numRows START AT offset	
Insert IGNORE	INSERT ... ON EXISTING SKIP	
Replace ...	INSERT ... ON EXISTING UPDATE	
FROM_DAYS ()	DAYS ()	
TO_DAYS ()	DATEADD (day, ...)	
WEEKDAY ()	DOW ()	
GROUP_CONCAT	LIST	
STD	STDDEV	
CHARACTER_LENGTH	LENGTH () function	
Position ()	LOCATE () function	
LOCALTIME, LOCALTIMESTAMP	NOW () built in function	
DECODE	CASE statement	
INSERT INTO . . . EFAULT VALUES.	INSERT INTO . . . VALUES (DEFAULT)	
LOAD DATA INFILE	LOAD TABLE	

Miscellaneous Syntax

The following is a miscellaneous list of compatibility items that do not fit into the aforementioned categories. It also includes mappings between functions that are not exactly the same, but are designed to provide the same functionality.

MySQL	**Sybase ASE**	**Comments**
VERSION ()	@@ version global variable	
mysql_insert_id ()	@@ identity global variable	
LAST_INSERT_ID variable	@@ identity global variable	
mysql_affected_rows ()	@@ rowcount global variable	
ANALYZE TABLE	sa_table_page_usage, sa_-table_fragmentation	Adaptive Server Enterprise also offers access to other properties via the **property** () function.
OPTIMIZE TABLE	CREATE STATISTICS	Adaptive Server Enterprise has a self-tuning optimizer that automatically maintains statistics, so statistics do not need to be updated manually.
CHECK TABLE	sa_validate () procedure	
USE *database-name*		There is no equivalent in Adaptive Server Enterprise. Each database running on a server requires its own connection.
LOCK TABLES (name) WRITE	LOCK TABLE table-name IN EXCLUSIVE MODE	Adaptive Server Enterprise supports row-level locking, so table locks are generally not required.
UNLOCK TABLES	COMMIT	A COMMIT releases all locks, unless a cursor is opened using the WITH HOLD clause.
Create table (KEY...)	CREATE TABLE ... CREATE INDEX	Adaptive Server requires two statements.
DO	CALL	
FLUSH/RESET	sa_flush_cache sa_flush_statistics	Most of the other flushable elements in MySQL are automatically managed by Adaptive Server Enterprise and do not need to be flushed.
expr1 SOUNDS LIKE expr2	SOUNDEX(expr1) = SOUNDEX(expr2)	
REGEX/RLIKE	SIMILAR	SIMILAR works differently from the mysql REGEX syntax, but performs the same function. It may suit the needs where the MySQL REGEXP expression is being used.
BINARY str	CAST str AS BINARY	
CURDATE () | CURRENT_DATE ()	CURRENT DATE	
CURTIME () | CURRENT_TIME ()	CURRENT TIME	

MySQL	Sybase ASE	Comments
SYSDATE () | LOCALTIME () | CURRENT_TIMESTAMP ()	CURRENT TIMESTAMP	
UTC_DATE ()	CURRENT UTC TIMESTAMP	
DATABASE ()	CURRENT DATABASE	
LOAD_FILE (file)	xp_read_file (file)	In Adaptive Server, the contents of file are returned as a long binary field, while in MySQL they are returned as a string.
CONNECTION_ID ()	CONNECTION_-PROPERTY ('Number')	

Other migration issues:

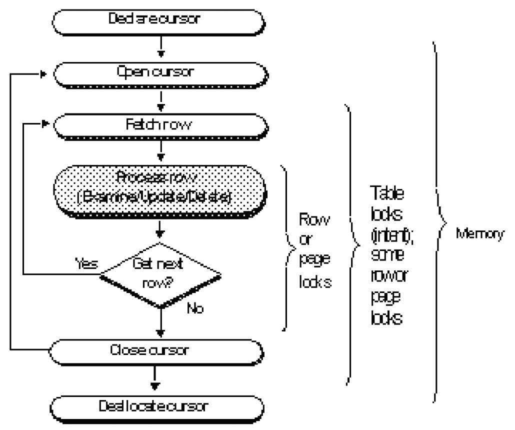
The following is a list of miscellaneous notes to keep in mind while migrating from MySQL to Sybase Adaptive Server Enterprise.

- The identifiers in MySQL are optionally enclosed with the back quote ('), while ASE uses the double quote (") or alternatively, square brackets ([]).
- Some words are keywords in ASE and not in MySQL such as comment and session.
- The minimum timestamp value in ASE is '0001-01-01 00:00:00', while it is '0000-00-00 00:00:00' in MySQL.
- Timestamps in MySQL have the format of YYYY-MM-DD hh:mm:ss. Sybase ASE includes fractions of a second as a part of timestamp value. The TIMESTAMP_FORMAT option allows you to specify the exact format used to return datetime values.
- While MySQL allows the use of single or double quotes around string literals, by default single quotes must be used to enclose string values in ASE. As previously mentioned, by default, double quotes signify the use of a database object identifier. This behavior can be changed by setting the QUOTED_IDENTIFIER option in the database.

APPENDIX F – ISSUES WITH THE USE OF CURSORS

Cursors are mechanisms for accessing the results of a SQL select statement one row at a time (or several rows, if you use **set cursors rows**). Since cursors use a different model from ordinary set-oriented SQL, the way cursors use memory and hold locks has performance implications for your applications. In particular, cursor performance issues includes locking at the page and at the table level, network resources, and overhead of processing instructions.

Cursors use memory and require locks on tables, data pages, and index pages. When you open a cursor, memory is allocated to the cursor and to store the query plan that is generated. While the cursor is open, Adaptive Server holds intent table locks and sometimes row or page locks. Following figure shows the duration of locks during cursor operations:



Resource used by cursor statement

In addition to locking, cursors involve more network activity than set operations and incur the overhead of processing instructions. The application program needs to communicate with Adaptive Server regarding every result row of the query.

Cursor performance issues include:

- Locking at the page and table level
- Network resources
- Overhead of processing instructions

Cursors on partitioned Heap Table:

A cursor scan of an unpartitioned heap table can read all data up to and including the final insertion made to that table, even if insertions took place after the cursor scan started.

If a heap table is partitioned, data can be inserted into one of the many page chains. The physical insertion point may be before or after the current position of a cursor scan. This means that a cursor scan against a partitioned table is *not* guaranteed to scan the final insertions made to that table.

If your cursor operations require all inserts to be made at the end of a single page chain, *do not* partition the table used in the cursor scan.

Optimizing tips for cursors

Here are several optimizing tips for cursors:

Optimize cursor selects using the cursor, not an ad hoc query:

1. A standalone **select** statement may be optimized very differently than the same **select** statement in an implicitly or explicitly updatable cursor. When you are developing applications that use cursors, always check your query plans and I/O statistics using the cursor, rather than using a standalone **select**. In particular, index restrictions of updatable cursors require very different access methods.

2. Use **union** or **union all** instead of **or** clauses or **in** lists:
   Cursors cannot use the dynamic index of row IDs generated by the OR strategy. Queries that use the OR strategy in standalone **select** statements usually perform table scans using read-only cursors. Updatable cursors may need to use a unique index and still require access to each data row, in sequence, in order to evaluate the query clauses. A read-only cursor using **union** creates a worktable when the cursor is declared, and sorts it to remove duplicates. Fetches are performed on the worktable. A cursor using **union all** can return duplicates and does not require a worktable.
3. Declare the cursor's intent:
   Always declare a cursor's intent: read-only or updatable. This gives you greater control over concurrency implications. If you do not specify the intent, Adaptive Server decides for you, and very often it chooses updatable cursors. Updatable cursors use update locks, thereby preventing other update locks or exclusive locks. If the update changes an indexed column, the optimizer may need to choose a table scan for the query, resulting in potentially difficult concurrency problems. Be sure to examine the query plans for queries that use updatable cursors.
4. Specify column names in the **for update** clause:
   Adaptive Server acquires update locks on the pages or rows of all tables that have columns listed in the **for update** clause of the cursor **select** statement. If the **for update** clause is not included in the cursor declaration, all tables referenced in the **from** clause acquire update locks.
5. Fetch more than one row if you are returning rows to the client.
   The SQL standard specifies a one-row fetch for cursors, which wastes network bandwidth. Using the **set cursor rows** query option and Open Client's transparent buffering of fetches, you can improve performance.

APPENDIX G – SQL TUNING & PERFORMANCE CONSIDERATIONS

Most of the gains in performance derive from good database design, thorough query analysis, and appropriate indexing.

When examining the application SQL the following should be considered/changed:

- If only a few rows are returned for queries it may be possible to create a non-clustered index on the table. This is called a covered query. For tables in the allpage-locking scheme this means that only the index pages are read.
- Consider index hints when table scans are used. This may typically be the case where the table data distribution is changed whilst running the application.
- Try to keep index entries as small as possible. You can create indexes with keys up to 600 bytes, but those indexes can store very few rows per index page, which increases the amount of disk I/O needed during queries. The index has more levels, and each level has more pages.
- If possible, do not include frequently updated columns as keys in clustered indexes on allpages-locked tables. When the keys are updated, the rows must be moved from the current location to a new page. Also, if the index is clustered, but not unique, updates are done in deferred mode.
- Good choices for clustered index keys are columns used in **order by** clauses and in joins.
- If your environment requires a lot of inserts, do not place the clustered index key on a steadily increasing value such as an IDENTITY column.
  Choose a key that places inserts on random pages to minimize lock contention while remaining useful in many queries. Often, the primary key does not meet this condition.
  This problem is less severe on data-only-locked tables, but is a major source of lock contention on allpages-locked tables.
- Most allpages-locked tables should have clustered indexes or use partitions to reduce contention on the last page of heaps.
  In a high-transaction environment, the locking on the last page severely limits throughput.
- If an index key is unique, define it as unique so the optimizer knows immediately that only one row matches a search argument or a join on the key.
- Be sure that the datatypes of the join columns in different tables are compatible. If Adaptive Server has to convert a datatype on one side of a join, it may not use an index for that table.

- Drop indexes that hurt performance. If an application performs data modifications during the day and generates reports at night, you may want to drop some indexes in the morning and re-create them at night.
- You might drop non-clustered indexes prior to a major set of inserts, and then rebuild them afterwards. In that way, the inserts and bulk copies go faster, since the non-clustered indexes do not have to be updated with every insert.
- Do not create nonclustered indexes, then clustered indexes. When you create the clustered index all previous nonclustered indexes are rebuilt.
- When deciding how many indexes to use, consider:
  - Space constraints
  - Access paths to table
  - Percentage of data modifications versus select operations
  - Performance requirements of reports versus OLTP
  - Performance impacts of index changes
  - How often you can use **update statistics**
- Add statistics for non-index columns. If non-index columns are used in joins this may be helpful to determine the most optimal query plan.
- Add update statistics. This may typically be the case where the table data distribution is changed whilst running the application.
- In triggers the frequent use of the inserted and deleted tables can be expensive. A rewrite of the triggers where cursors are used and the inserted/deleted tables are only scanned once will be more efficient.
- The use of concatenated fields in the where clause will result in table scans because the Sybase cost optimizer will not be able to use the indexes. Rewrite of the SQL to have individual fields addressed in the where clause will be necessary.
- When using multiple table joins it may save parse and compile time when the most efficient order of tables is programmed and the set forceplan option used.
- Use stored procedures to reduce compilation time and network usage.
- Using the minimum locking level that meets your application needs
- Follow these guidelines when you write search arguments for your queries:
  - Avoid functions, arithmetic operations, and other expressions on the column side of search clauses. When possible, move functions and other operations to the expression side of the clause.
  - Avoid incompatible datatypes for columns that will be joined and for *variables* and parameter used as search arguments.
  - Use the leading column of a composite index as a search argument. The optimization of secondary keys provides less performance.
  - Use all the search arguments you can to give the optimizer as much as possible to work with.
  - If a query has more than 102 predicates for a table, put the most potentially useful clauses near the beginning of the query, since only the first 102 SARGs on each table are used during optimization. (All of the search conditions are used to qualify the rows.)
  - Some queries using > (greater than) may perform better if you can rewrite them to use >= (greater than or equal to). For example, this query, with an index on int_col uses the index to find the first value where int_col equals 3, and then scans forward to find the first value that is greater than 3. If there are many rows where int_col equals 3, the server has to scan many pages to find the first row where *int_col* is greater than 3:

```
select * from table1 where int_col > 3
```

  It is probably more efficient to write the query like this:

```
select * from table1 where int_col >= 4
```

  This optimization is more difficult with character strings and floating-point data. You need to know your data.
  - Check **showplan** output to see which keys and indexes are used.
  - If you expect an index is not being used when you expect it to be, check **dbcc traceon (302)** output to see if the optimizer is considering the index.

- Avoid using following search arguments in where clause of the query, as they cannot be optimized:

```
The following search arguments cannot be optimized:
advance * 2 = 5000  /*expression on column side
                                    not permitted */
substring (au_lname,1,3) = "Ben" /* function on
                                                column name */
```

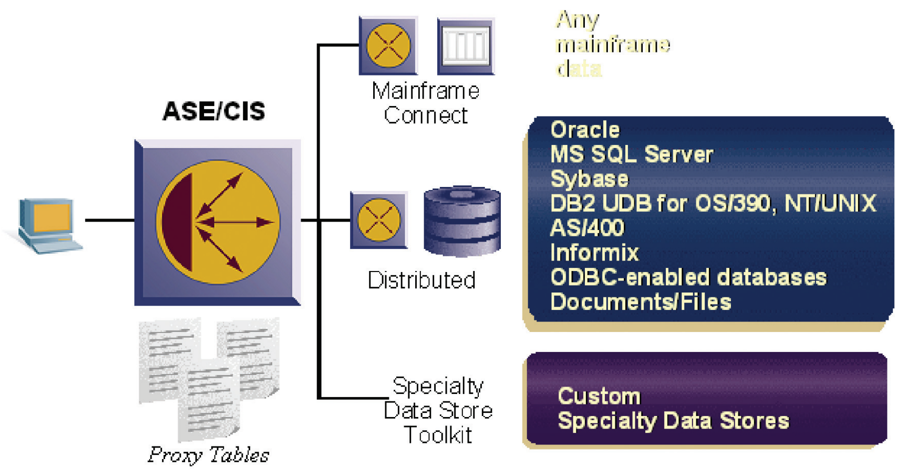
These two clauses can be optimized if written in this form:

```
advance = 5000/2
au_lname like "Ben%"
```

- **When using the UNION statement**, keep in mind that, by default, it performs the equivalent of a SELECT DISTINCT on the final result set. In other words, UNION takes the results of two like record sets, combines them, and then performs a SELECT DISTINCT in order to eliminate any duplicate rows. This process occurs even if there are no duplicate records in the final record-set. If you know that there are duplicate records, and this presents a problem for your application, then by all means use the UNION statement to eliminate the duplicate rows.
  On the other hand, if you know that there will never be any duplicate rows, or if there are, and this presents no problem to your application, then you should use the UNION ALL statement instead of the UNION statement. The advantage of the UNION ALL is that is does not perform the SELECT DISTINCT function, which saves a lot of unnecessary Server resources from being using.
- **Carefully evaluate whether your SELECT query needs the DISTINCT clause or not.** Do not add this clause to every one of SELECT statements, even when it is not necessary.
  The DISTINCT clause should only be used in SELECT statements if you know that duplicate returned rows are a possibility, and that having duplicate rows in the result set would cause problems with your application.
  The DISTINCT clause creates a lot of extra work for Server, and reduces the physical resources that other SQL statements have at their disposal. Because of this, only use the DISTINCT clause if it is necessary.
- **If you need to verify the existence of a record in a table**, don't use SELECT COUNT (*) in your Transact-SQL code to identify it, which is very inefficient and wastes server resources. Instead, use the Transact-SQL IF EXITS to determine if the record in question exits, which is much more efficient.
- **If the number rows returned by a query is large**, using a clustered index and non-clustered index can cost more than table scan.

APPENDIX H – USING CIS AND ENTERPRISE CONNECT DATA ACCESS

Using Component Integration Services, together with Enterprise Connect Data Access Option for MySQL, enables the transparent access to the MySQL database with the ASE. This allows accessing both Sybase and MySQL database the same time. So you transfer the data of the MySQL table into a table on Sybase server by means of a select into statement. Component Integration Services manages the data regardless of the location of the external servers.



You must configure Component Integration Services, before accessing the data on the remote server.

- Add the remote server to the sql.ini file or interfaces file.
- Add the name, server class, and network name of the remote server to system tables by using sp_addserver system stored procedure.
- Optionally, assign an alternate login name and password for communicating with a remote server.

ECDA provides the following Sybase Central plug-ins:

- Adaptive Server Enterprise plug-in, which allows you to manage and monitor Adaptive Server installations. This plug-in also allows you to set up proxy tables with ASE/CIS for access to remote, non-Sybase data sources.
- DirectConnect plug-in (DirectConnect Manager), which allows you to manage DirectConnect servers locally or remotely throughout your business environment.

Using ECDA you can share data efficiently throughout your organization by moving it in bulk form between data sources using the DirectConnect transfer feature or the Adaptive Server select into statement.

ECDA data access servers provide connectivity between clients located throughout the enterprise and enterprise data servers. So, you can use a DirectConnect server on its own to connect directly to a MySQL data server. Or, you can combine DirectConnect servers with Adaptive Server, the Sybase products that provide a Transact-SQL™ interface so you can access and transfer data in a heterogeneous environment.

To transfer the data from MySQL to ASE, it requires following:

- ECDA for MySQL Server
- Sybase ASE ODBC driver

For more detail about how to transfer the data from MySQL to ASE, refer User's Guide for Enterprise Connect Data Access Options.

APPENDIX I – USING POWERDESIGNER

PowerDesigner has a powerful reverse engineering capability that can import a complete schema into a physical data model automatically. Once the schema is imported, change the data base type of the model to Adaptive Server Enterprise 15.0. PowerDesigner will convert the schema to use Sybase ASE data types. The automatically chosen mappings should be correct, however the schema should be reviewed to make sure they make sense for your application. You may choose to change a data type in the target schema to a different, but similar data type than is in the source.

Steps to reverse engineer the schema (tables/indexes/views/constraints):

- Start PowerDesigner
- Choose >File>Reverse Engineer>Database to open the new Physical Data Model dialog Box.
- Select the MySQL database from the list and click on OK
- When the Database Reverse Engineering dialog box opens, click the Using an ODBC data source radio button
- Click the Connect to an ODBC Data Source tool to open the Connect to an ODBC Data Source dialog box.
- Select the Machine data source radio button and select MySQL data source from the list.
- Type a user ID and a password, and then click Connect. To return to the Database Reverse Engineering dialog box.
- Click OK to open the ODBC Reverse Engineering dialog box. This box allows you to specify a selection of objects to reverse engineer. Only tables and triggers are selected by default.
- Click OK to begin the process of reverse engineering. When the process is complete, a confirmation message is given in the Output window
- Choose the installed ODBC driver and connect
- Select all and create the PDM (Physical Data Model)
- Save the PDM with >file>save as (use the Sybase database name as name of file)

Before creating the schema script from a PDM, change the target DBMS to Adaptive Server 15.0 which converts the datatypes of the model.

Steps to change the target DBMS to ASE 15.0:

- Choose>Database>Change Current DBMS.
- Select a target DBMS as 'Sybase AS Enterprise 15.0' from the dropdown listbox.

Now, create the reverse engineering script from the PDM created.

- Choose database>Generate Database to open the Database Generation dialog box
- Type a destination directory and a filename for the script file in the Directory and File Name boxes.
- Select the Script generation radio button.
- The output window shows the progress of the generation process, and indicates the syntax for running the script. At the end of script generation, a Result box appears. It lists the file path of the generated script file. Click Edit to open the script in a text editor or Close to close the Result box.
- Check all warnings and errors, if needed make changes and generate script again
- Run the script and check for any errors.
- Make changes where needed and rerun script

The following cannot be automatically converted with the tool and will have to be changed manually (and possible workarounds needed). For recommendation of datatype conversion see appendix C and for language changes see appendix E:

- For functions used in constraints like len () in MySQL and char_length () in Sybase

APPENDIX J – USING BCP

Bulk copy utility (bcp) provided by is used to move data between Adaptive Server and operating system file.

bcp provides a convenient, high-speed method for transferring data between a database table or view and an operating system file. **bcp** can read or write files in a wide variety of formats. When copying in from a file, **bcp** inserts data into an existing database table; when copying out to a file, **bcp** overwrites any previous contents of the file.

A detailed description of **bcp** syntax:

```
bcp [[database_name.] owner.]table_name [: [ partition_id | slice_number]|
   partition partition_name] {in | out} datafile
   [--show-fi]
   [--hide-vcc]
   [-f formatfile]
   [-e errfile]
   [-F firstrow]
   [-L lastrow]
   [-b batchsize]
   [-m maxerrors]
   [-n]
   [-c]
   [-t field_terminator]
   [-r row_terminator]
   [-U username]
   [-P password]
   [-I interfaces_file]
   [-S server]
   [-a display_charset]
   [-z language]
   [-A packet_size]
```

```
[-J client_charset]
[-T text_or_image_size]
[-E]
[-g id_start_value]
[-N]
[-X]
[-K keytab_file]
[-R remote_server_principal]
[-V [security_options]]
[-Z security_mechanism]
[-Q]
[-Y]
[--maxconn maximum_connections]
```

Commonly used parameters:

database_name
is optional if the table being copied is in your default database or in master. Otherwise, you must specify a database name.

owner
is optional if you or the Database Owner owns the table being copied. If you do not specify an owner, bcp looks first for a table of that name that you own, and then looks for one owned by the Database Owner. If another user owns the table, you must specify the owner name or the command fails.

partition_id
specifies the partition number into which data is to be copied. It is supported only for bcp in.

partition partition_name
specifies a set of one or more partitions, separated by commas.

in | out
is the direction of the copy. in indicates a copy from a file into the database table; out indicates a copy to a file from the database table or view.

datafile
specifies a set of one or more unique data files, separated by commas. It is supported for both bcp in and bcp out. The path name can be from 1 to 255 characters in length.

-f formatfile
is the full path name of a file with stored responses from a previous use of bcp on the same table. After you answer bcp's format questions, it prompts you to save your answers in a format file. Creation of the format file is optional. The default file name is bcp.fmt. The bcp program can refer to a format file when you are copying data so that you do not have to duplicate your previous format responses interactively. Use the -f parameter only if you previously created a format file that you want to use now for a copy in or copy out. If you do not specify this parameter, bcp interactively queries you for format information.

-e errfile
is the full path name of an error file where bcp stores any rows that it was unable to transfer from the file to the database. Error messages from bcp appear on your terminal. bcp creates an error file only when you specify this parameter.

-c
performs the copy operation with char datatype as the default storage type of all columns in the data file. Use this format if you are sharing data between platforms. This parameter does not prompt for each field; it uses char as the default storage type, no prefixes, \t (tab) as the default field terminator, and \n (new line) as the default row terminator.

-t field_terminator
specifies the default field terminator.
-U username
specifies an Adaptive Server login name.
-P password
specifies an Adaptive Server password. If you do not specify -Ppassword, bcp prompts for a password. You can leave out the -P flag if your password is NULL.
-I interfaces_file
specifies the name and location of the interfaces file to search when connecting to Adaptive Server. If you do not specify -I, bcp looks for an interfaces file (sql.ini in Windows) located in the directory specified by the SYBASE environment variable (ini directory in Windows).
-S server
specifies the name of the Adaptive Server to which to connect. If you specify -S with no argument, bcp uses the server specified by the DSQUERY environment variable.

Importing and exporting data with *bcp*

Transact-SQL commands cannot transfer data in bulk. For this reason, use **bcp** for any large transfers. **bcp** can be used to:

- Import data that was previously associated with another program, such as the records from another database management system. This is the most common use for **bcp**.

  Before using **bcp**, you must create a file of the records you want to import. The general steps are:
  1. Put the data to transfer into an operating system file.
  2. Run **bcp** from the operating system command line.
- Move tables between Adaptive Servers or between Adaptive Server and other data sources that can produce an operating-system file.
- Transfer data for use with other programs, for example, with a spreadsheet program. The general steps to transfer data are:
  a. Use **bcp** to move the data from Adaptive Server into an operating-system file from which the other program imports the data.
  b. When you finish using your data with the other program, copy it into an operating-system file, and then use **bcp** to copy it into Adaptive Server.

Adaptive Server can accept data in any character or binary format, as long as the data file describes either the length of the fields or the terminators, the characters that separate columns.

The structures in the tables involved in the transfer need not be identical, because when **bcp**:

- Imports *from* a file, it appends data to an existing database table.
- Exports *to* a file, it overwrites the previous contents of the file.

When the transfer is complete, **bcp** informs you of the:

- Number of rows of data successfully copied
- Number of rows (if any) that it could not copy
- Total time the copy took.
- Average amount of time, in milliseconds, that it took to copy one row.
- Number of rows copied per second.

If **bcp** runs successfully, you see a return status of 0. The return status generally reflects errors from the operating system level and correspond to the ones listed in the *errno.h* file in the */usr/include/sys/* directory.

bcp requirements

Before using **bcp**, you need to provide it with basic data information and prepare both the data for transfer and the command to access the data.

You must supply the following information to transfer data successfully to and from Adaptive Server:

- Name of the database and table or view
- Name of the operating system file
- Direction of the transfer (**in** or **out**)

Pre-transfer tasks

Before you can use **bcp in**, you must prepare the command and the data for transfer:

- To use either fast or slow **bcp**, set database option **'into/bulkcopy/pllsort'** to **true**. For example, to turn on this option for the xyz database, you would enter:
```
sp_dboption, "select into/bulkcopy/pllsort", true
```
- To use fast **bcp**, remove indexes and triggers on the target table.

bcp modes

bcp in works in one of two modes:

- Slow **bcp** – logs each row insert that it makes, used for tables that have one or more indexes or triggers.
- Fast **bcp** – logs only page allocations, copying data into tables without indexes or triggers at the fastest speed possible.

To determine the **bcp** mode that is best for your copying task, consider the:

- Size of the table into which you are copying data
- Amount of data that you are copying in
- Number of indexes on the table
- Amount of spare database device space that you have for re-creating indexes

bcp performance

Keeping indexes and triggers on a table causes the bulk copy utility to use slow **bcp** automatically. However, slow **bcp** can fill the transaction log very quickly.

- When you are copying a large number of rows, the performance penalty and log space requirements for using slow **bcp** can be severe.
- For extremely large tables, using slow **bcp** is not an option because its detailed log makes it much too slow.

To improve the performance of **bcp**:

- Use partitioned tables. Several bcp sessions with a partitioned table can reduce dramatically the time required to copy the data. However, such performance improvements are more noticeable in fast **bcp** than in slow **bcp**.

Use **bcp** in parallel to increase performance dramatically. Parallel bulk copy can provide balanced data distribution across partitions. For more information, see *"Using parallel bulk copy to copy data into a specific partition"*.

APPENDIX K – USING POWERTRANSFER

PowerTransfer is an extension to PowerDesigner that allows the transfer of data into Adaptive Server. The source database can be any ODBC data source supported by PowerDesigner. PowerTransfer uses the Sybase bulk copy mechanism (**bcp**) for high-performance inserts into Adaptive Server.

Power Transfer allows data to be transferred from a wide variety of sources into Sybase Adaptive Server Enterprise. It uses ODBC to pull data from a source database and Sybase Bulk Library to insert it into the target.

In order to migrate the MySQL database to ASE, a database administrator would use PowerDesigner® to migrate the schema, followed by Power Transfer to copy over the data. The schema can be migrated using any method, however PowerDesigner is well suited for this task.

MIGRATING THE SCHEMA

The first step is to migrate the schema from the MySQL database to the Sybase database. This can be done by Power Designer mentioned in Appendix I.

Power Transfer can handle most conversions between types, however be aware that overflows may occur during the data transfer if an inappropriate conversion is selected. Power Transfer will allow you to change the names of columns in the target schema but you may not change the column ordering. The table names must be the same in the source and target database.

Once the changes to the physical model are complete, use PowerDesigner to connect to the ASE database and push the schema out to it. It is recommended that the schema be pushed in two steps. Before the data is transferred, the target database should contain table definitions without indexes, triggers, referential integrity rules, constraints, or defaults. The presence of these items will slow the bulk insert process. Additionally, complicated referential integrity rules may make it difficult to move the data in the correct order that is required. After the data is transferred, use PowerDesigner to modify the target database by adding in these items.

COPYING THE DATA

Once the database schema is present on the target server, you can use Power Transfer to copy the data. Before starting Power Transfer, Open Client should be configured with an interfaces file entry for the target server. Power Transfer uses Sybase Bulk Library to insert data into ASE. Bulk Library allows data to streamed in large chunks giving significantly better performance than generating INSERT statements. The target ASE database should also be configured to allow bulk operations. This can be done through the following command:

```
sp_dboption databasename,"select into", true
```

The first screen you will be presented with is the Server Login screen. From here, choose the ODBC data source and the target Adaptive Server Enterprise. Fill in the login information and choose connect.

Various transfer options can be selected before starting the transfer:

- **Rows Per Bulk Insert**
  This value determines the number of rows that Power Transfer sends in a single bulk request. It corresponds to the array binding size in bulk library. Larger values will yield better performance by reducing network overhead, however this is probably not a large factor unless very small rows are being sent. The default of 100 is reasonable for most situations.
- **Rows Per Commit**
  This value determines the frequency that Adaptive Server Enterprise will commit the bulk-inserted data to disk. Performance will be better as this value is increased, but the ASE transaction log could fill up if the value is too large. Adjust this value up or down depending on the capacity of transaction log in the target ASE.
- **Number of Connections**
  Power Transfer can open more than one connection to the source and target database so that transfers are performed in parallel. This option can significantly improve performance, however if chosen, it is important that rules and triggers related to referential integrity not be present in the target database. When parallel transfers are selected, the order that the tables are transferred in is no longer guaranteed to be the order listed in the table copy screen. On a single CPU, a recommendation would be to use 6 connections. Assuming a server class machine is being used for ASE, this value should be increased until the Power Transfer machine is completely CPU bound.
- **Packet Size**
  Increasing the packet size that Power Transfer uses to connect to ASE will significantly improve performance. This value must be less than or equal to the 'max network packet size' values set in sp_configure. It is recommended that this value be set to 8K or better.
- **Truncate Target Table Before Copy**
  Enabling this option causes a truncate table command to be issued to the target prior to the transfer.

- **Show Views In Table List**
  This option will allow you to transfer the results of a view in a source database into a table in the target ASE. This option is useful if you do not want to copy the entire contents of the original source table. A view could be set up that omits or reorders columns, or aggregates data from multiple tables.
- **Use Owner Names**
  Selecting this option will cause tables names to be fully qualified using the owner name. If this is not selected, owner names will not be used when referencing tables.

The Table Copy screen will show the list of tables that are located in the source ODBC database. Either select tables in the Source Tables list individually or choose "Select All" if you want to choose all tables. Then choose "Add" to queue up the selected tables for transfer. If you are using one connection, the tables will be transferred sequentially in the order shown in the "Tables To Be Copied" box. Select *Start Copy* to initiate the transfer.

Performance of Power Transfer depends on three important factors: ASE host speed, Power Transfer host speed, and the speed of the network. Assuming the source and target database are running on server class machines, the performance limitation is most likely to be the CPU speed of the Power Transfer machine.

APPENDIX L – USING INFOMAKER

The Data Pipeline painter feature of InfoMaker gives you the ability to reproduce data quickly within a database, across databases, or even across DBMSs. To do that, you create a data pipeline, which, when executed, pipes the data as specified in the definition of the data pipeline.

With the Data Pipeline painter, you can perform following tasks:

- Pipe data (and extended attributes) from one or more tables to a table in the same DBMS or a different DBMS
- Pipe an entire database, a table at a time, to another DBMS (and if needed, pipe the database's extended attribute system tables)
- Create a table with the same design as an existing table but with no data

When you pipe data from one MySQL to Sybase, InfoMaker makes a best guess at the appropriate destination datatypes. You can correct InfoMaker's best guess in your pipeline definition as needed. For more details about how to pipe data from MySQL to ASE, see the user's Guide of InfoMaker for Data Pipeline painter.

SYBASE®