

ORACLE TO SAP SYBASE ASE MIGRATION GUIDE

BY PETER DOBLER
2013



1 TABLE OF CONTENTS

2	Preface	10
2.1	Intended Audience	10
2.2	What You Should Already Know	10
2.3	Related Documents	10
3	Overview	11
3.1	Introduction	11
4	Migration Process	12
4.1	Application Components	12
4.2	Migrating the Data	13
4.3	Migrating the Application	13
4.4	Validate Migration	14
5	Oracle and Sybase ASE Compared	15
5.1	Architecture	16
5.1.1	Databases and Instances	18
5.1.2	Connecting to Oracle	19
5.2	Memory Structure	20
5.2.1	Sybase ASE allocates memory as follows:	21
5.3	Processes	23
5.3.1	Oracle Process Management	23
5.3.2	Sybase Process Management	25
5.4	Concurrency and Multi-Versioning	26
5.4.1	ANSI Isolation Levels	27
5.5	Schema Migration	30
5.5.1	Schema Object Similarities	30
5.5.2	Schema Object Names	31

5.5.3	Table Design Considerations.....	31
5.6	Data Types	34
5.7	Data Storage Concepts.....	36
5.7.1	Datafiles	36
5.7.2	Control Files.....	37
5.7.3	Redo Log Files.....	37
5.7.4	Data Storage Concept Mapping	38
5.7.5	Overview of the main components of the Oracle Database.....	42
5.7.6	Overview of the main components of the Sybase ASE Server	42
5.8	Data Manipulation Language	46
5.8.1	SQL Language.....	46
5.8.2	Conceptual SQL Language Differences.....	46
5.8.3	Database Security.....	57
5.8.4	Transaction Processing in Stored Procedures.....	60
6	DBA Operations Guide	62
6.1	Transactions	62
6.2	Redo and Undo	63
6.2.1	Redo Log.....	63
6.2.2	Undo.....	64
6.3	Tables and Indexes	65
6.3.1	Tables.....	65
6.3.2	Indexes	66
6.4	System Views	68
6.4.1	Oracle Static Data Dictionary Views.....	68
6.4.2	Oracle Dynamic Performance Views.....	68
6.4.3	Sybase ASE Static Data Dictionary Tables.....	69
6.4.4	Sybase ASE Dynamic Performance Tables	70

6.4.5	Static Data Dictionary Comparison	70
6.4.6	Dynamic Performance Views Comparison	73
6.5	Parallel Execution	74
6.6	Data Loading and Unloading	79
6.6.1	Data Loading	79
6.6.2	Data Unloading	82
6.7	Migrate the Database Users	83
6.7.1	Create User Account	84
6.7.2	Create Roles and privileges	85
6.8	Backup and Restore	88
6.9	Other Administrator Tasks	92
6.9.1	Database Consistency Checks	92
6.9.2	Threshold checking	93
6.9.3	Reorganizing table and index space	93
6.9.4	Refreshing the table statistics	94
6.9.5	Monitor space usage	94
6.10	Oracle DBA Tools and Sybase DBA Tools Compared	94
7	Migrating the Database Architecture	99
7.1	Build the Sybase Instance	99
7.1.1	Pre installing planning	100
7.1.2	Installation	101
7.2	Configure the Server	102
7.2.1	Server level configuration:	103
7.2.2	Database Level configuration:	106
7.3	Migrate the Storage Architecture	107
7.3.1	Database Device or Logical Device:	108
7.3.2	Database:	109

7.3.3	Segments:.....	110
7.3.4	Partition:.....	111
7.3.5	Example:.....	112
7.3.6	Improve performance with Object placement:.....	113
8	Migrating the Data and SQL.....	114
8.1	Factors in Migration.....	114
8.2	Options for Migration	114
8.2.1	PowerDesigner	115
8.2.2	Sybase Bulk Copy.....	115
8.2.3	Component Integrated Services	116
8.2.4	Enterprise Connect Data Access Option for Oracle.....	117
8.3	Migrating the database objects and SQL application Code	119
8.3.1	Migrating Database Objects.....	119
8.3.2	Migrating the SQL application code.....	123
8.4	Validate the Data Migration	125
8.4.1	Verify the Data Transfer.....	125
8.4.2	Validate the Data Integrity	125
9	Application Migration	126
9.1	Embedded SQL Application	126
9.2	ODBC or JDBC Client Application	129
9.2.1	ODBC Client Application.....	129
9.2.2	JDBC Client Application	130
9.2.3	Oracle JDBC Connection	131
9.2.4	Sybase JDBC Connection.....	131
9.3	Database-Specific Client Library Application.....	132
9.4	C Applications.....	133
9.4.1	Error handling in Ct-Library	136

9.5	Oracle Forms	139
9.5.1	Migration Approach	139
9.5.2	Evaluating Oracle Forms	140
9.5.3	Understanding PowerBuilder.....	142
9.5.4	Validation of PowerBuilder Application	145
9.6	Triggers and Stored Procedures	146
9.6.1	Triggers	146
9.6.2	Stored Procedures.....	147
9.6.3	Data Types.....	150
9.6.4	Schema Objects.....	152
9.6.5	PL/SQL vs. T/SQL Constructs	161
9.6.6	PL/SQL vs. T/SQL Language Elements	178
10	Validate Database Server Migration.....	183
10.1	Physical architecture of the database	183
10.2	Security	184
10.3	Logical architecture of the database.....	184
10.4	Data	184
10.5	Functionality	184
10.6	Performance.....	184
11	Sybase ASE Features	185
11.1	Additional ASE Features.....	185
11.1.1	Locking Scheme	185
11.1.2	Memory Management	187
11.1.3	Logical Process Manager	188
11.1.4	User Defined Roles	188
11.1.5	DBCC.....	189
11.1.6	Parallel Query	191

11.1.7	Backup Server.....	192
11.1.8	ASE with Encrypted Columns.....	192
11.1.9	Password Protected Backups	194
11.1.10	Enhanced Full-Text Search Specialty Data Store.....	194
11.1.11	Historical Server.....	196
11.1.12	Scrollable Cursors	199
11.1.13	Automatic Update Statistics	200
11.1.14	Multiple Temporary Database	200
11.2	Monitoring Tools	202
11.2.1	Monitoring the Performance of the Queries.....	202
11.2.2	Monitor Concurrency	202
11.2.3	Monitor the ASE Server.....	203
11.3	Job Scheduler	205
11.4	Sybase Control Center	207
11.4.1	Alerts	208
11.4.2	Repository	208
11.4.3	Logging	209
11.4.4	Heat Chart.....	209
11.4.5	Historical Performance Monitoring.....	210
11.4.6	Manage and Monitor a Sybase ASE Server Environment.....	210
12	APPENDIX A – Portability Check.....	212
12.1	Workarounds for Sybase ASE	212
12.2	Oracle to Sybase ASE Incompatibilities.....	219
12.3	Search for Keywords in Oracle Source	222
13	APPENDIX B – Migration Checklist.....	223
14	APPENDIX C – Data Type Differences	226
15	APPENDIX D – SQL Language Differences	228

15.1	General	228
15.2	Operators.....	233
15.3	Functions.....	234
15.3.1	System Functions.....	234
15.3.2	Data Functions.....	234
15.3.3	Conditional functions.....	235
15.4	Joins	237
15.5	Cursors.....	237
15.6	Control of flow language.....	238
15.7	Error Handling	239
15.8	Examples of the differences in SQL.....	241
15.8.1	String Manipulation	241
15.8.2	Numeric Manipulation.....	241
15.8.3	Control Structures	241
15.8.4	Stored procedure.....	242
15.8.5	Date Manipulations.....	242
15.8.6	Miscellaneous.....	244
15.8.7	Outer join Syntax	244
15.8.8	Limitations.....	245
16	APPENDIX E – Issues With Use of Cursors.....	246
17	APPENDIX F – SQL Tuning & Performance Considerations.....	250
18	APPENDIX G – Using 3 rd Party Tools	254
18.1	PowerDesigner.....	254
18.2	SwisSQL.....	257
18.3	FACT	257
18.4	ETL Tools.....	258
19	APPENDIX H – Using BCP	259

20	APPENDIX I – Step by Step Migration Example.....	265
20.1	Planning the Migration	265
20.2	Installing the Software.....	266
20.2.1	SySAM	266
20.2.2	Pre-Installation.....	267
20.2.3	Installation.....	267
20.3	Migrating the User Logins.....	268
20.4	Migrating the Schema	269
20.5	Mapping the Data Types	271
20.6	Unloading and Loading the Data	272
20.7	Setup the Backup	276
20.8	Setup Maintenance Tasks.....	283
20.9	Setup Monitoring	289
20.9.1	Oracle	289
20.9.2	Sybase ASE	291
20.10	Migrating Triggers and Stored Procedures.....	292
20.10.1	SQL Hints.....	293
20.10.2	Join Orders	296
20.11	Migrating Application Connections through JDBC.....	297
20.11.1	Oracle JDBC Extended Data Sources	298
20.11.2	Sybase ASE JDBC.....	299
20.11.3	JDBC Standard Data Sources	300
20.12	Migrating SQL Code inside the Application.....	301
21	Index	302

2 PREFACE

The Oracle to Sybase Migration Guide provides detailed information on migrating a database from Oracle® 9i, 10g and 11g to Sybase Adaptive Server® Enterprise (ASE). The processes and differences outlined in this guide will help you to manage the migration process, regardless the conversion tool you are using to perform the migration. The focus is on the Oracle to Sybase ASE migration and Oracle features like data replication or data warehousing are better utilized in other specialized Sybase products.

If not otherwise specified all stand-alone references to simply Sybase ASE or Adaptive Server are considered references to Sybase Adaptive Server® Enterprise.

2.1 INTENDED AUDIENCE

This guide is intended for anyone who is involved in converting an Oracle® database to Sybase Adaptive Server® Enterprise (ASE).

2.2 WHAT YOU SHOULD ALREADY KNOW

You should be familiar with relational database concepts and with the operating system environments under which you are running Oracle and Sybase ASE.

2.3 RELATED DOCUMENTS

For more detailed information about the Sybase ASE tools and functions see these user manuals and white papers:

- Sybase ASE Product Manuals at <http://infocenter.sybase.com>
- MDA Table White Papers and TechWave presentations at <http://sybase.com>

3 OVERVIEW

3.1 INTRODUCTION

This guide assists with the migration process from Oracle® to Sybase Adaptive Server® Enterprise (ASE). By migration we mean the process of changing an application so that it uses Sybase ASE, rather than the Oracle RDBMS, as its underlying database management system. This guide is organized into several distinct chapters. There is a general architecture and language comparison chapter, a chapter for database management specific comparisons and guidelines, and an application migration specific chapter. The example application migration is aimed at E-commerce applications implemented in a 3-tier architecture. Web/Java® considerations will be addressed. Non-Web/Java based application can skip the sections concerning these issues.

The main differences between the RDBMS for Sybase ASE and Oracle will be explained and introduces the main Sybase ASE features needed from the point of view of an application designer/developer and DBA. It describes a systematic process for migrating the application including migrating architecture, data, SQL, and client application. It also describes the validation steps for the successful migration of the database. Checklists to assist with the migration are presented as appendices.

This document assumes the user has a DBA level understanding of an Oracle database and is in the process of becoming familiar with a Sybase ASE database.

The top issues when converting the application that are the most time consuming are:

1. Conversion of the application when nonstandard SQL-92 SQL language extensions are used.
2. Conversion of the application when business logic and database access code is mixed in a non-modular design.
3. Conversion of the database when non-Sybase convertible data types are used and when Sybase ASE limitations are exceeded.
4. Migrating Oracle features that are not supported in Sybase ASE using a workaround approach.

4 MIGRATION PROCESS

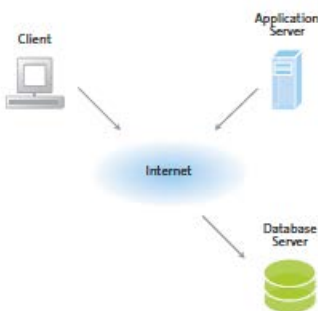
When migrating an Oracle application all the application components have to be checked for specific Sybase ASE issues. The application may contain Oracle specific features.

The migration process can be split into five main parts:

- Schema and Data Migration
- SQL Conversion – Server and Client side
- Application Data Access Conversion
- Application Validation
- Maintenance and Administration Tasks

4.1 APPLICATION COMPONENTS

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



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

4.2 MIGRATING THE DATA

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

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

The recommended tools for the migration are:

- PowerDesigner® for the data schema creation
- BCP (Sybase Bulk Copy) for moving high volume data
- CIS (Sybase Component Integration Services) for moving the data
- Sybase Replication Server with ExpressConnect for Oracle for real time data movement

Appendix C describes the recommended data type conversion and Appendix D contains the data schema details that the recommended tools cannot convert automatically.

4.3 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 Oracle specific SQL. Stored procedures and triggers have to be checked for SQL language differences described in the section Stored Procedures and Triggers as well as in Appendix D. At this time also consider the tuning and performance considerations described in Appendix E.

4.4 VALIDATE MIGRATION

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

The testing should include validation of:

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

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

5 ORACLE AND SYBASE ASE COMPARED

Before deep diving into the migration steps for DBAs and developers, it is necessary to understand the underlying differences between Oracle and Sybase ASE. This knowledge will guide you through the migration process and is the base for a successful migration.

One important fact about comparing Oracle with Sybase ASE is to understand which part of Oracle will be compared with which Sybase product. Unlike Oracle, which integrates all its technology into one system, Sybase opted to separate these tasks with unique and specialized products that communicated well with each other.

The following list illustrates how Oracle features compare to Sybase products. This is a subset of Oracle features and Sybase products, but outlines how Oracle and Sybase can be aligned.

Oracle	Sybase
Oracle Database Server	Sybase ASE
Oracle OLAP and DW	Sybase IQ
Oracle Analytics	Sybase IQ
Oracle RAC	Sybase ASE Cluster Edition
Oracle Times Ten	Sybase ASE 15.5 In-Memory Database
Oracle Data Guard	Sybase Mirror Activator
Oracle Streams	Sybase Replication Server
Oracle ExaData	Sybase IQ Appliance
MySQL	Sybase ASE 15 Express, Free to use on Linux.

In the interest of keeping this guide usable and pointed, we will focus on the migration from Oracle Database Server to Sybase ASE. All the other Oracle features and how they would migrate to a Sybase product will be subject to another guide.

Appendix A contains a portability checklist that outlines the Oracle features not supported by Sybase ASE and what to look out for.

Appendix B contains a checklist on what preparations need to be performed to ensure that a migration is actually possible.

5.1 ARCHITECTURE

The physical architecture of Oracle looks different on different operating systems. For example, on a UNIX operating system you will see Oracle implemented as many different operating system processes, with virtually a process per major function. On the Windows platform, Oracle is implemented as a single, threaded process. This is the architecture Sybase ASE chooses for all platforms. Regardless the operating system, Sybase ASE consists of one single, threaded process, at least for the database core application. Unlike Oracle, Sybase ASE separates near core application functions into separate single processes that communicate via the Component Integration Service (CIS) with each other.

Typical Oracle Processes on Linux:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
oracle	6410	1	0	Feb08	?	00:03:38	ora_pmon_ORCL
oracle	6412	1	0	Feb08	?	00:00:17	ora_vktm_ORCL
oracle	6416	1	0	Feb08	?	00:00:05	ora_diag_ORCL
oracle	6418	1	0	Feb08	?	00:00:13	ora_dbrm_ORCL
oracle	6420	1	0	Feb08	?	00:02:33	ora_psp0_ORCL
oracle	6424	1	0	Feb08	?	00:00:56	ora_dia0_ORCL
oracle	6426	1	0	Feb08	?	00:00:29	ora_mman_ORCL
oracle	6428	1	0	Feb08	?	00:02:52	ora_dbw0_ORCL
oracle	6430	1	0	Feb08	?	00:07:54	ora_lgwr_ORCL
oracle	6432	1	0	Feb08	?	00:07:05	ora_ckpt_ORCL
oracle	6434	1	0	Feb08	?	00:10:44	ora_smon_ORCL
oracle	6436	1	0	Feb08	?	00:00:03	ora_reco_ORCL
oracle	6438	1	0	Feb08	?	00:06:23	ora_mmon_ORCL
oracle	6440	1	0	Feb08	?	00:02:43	ora_mmm1_ORCL
oracle	6442	1	0	Feb08	?	00:00:01	ora_d000_ORCL
oracle	6444	1	0	Feb08	?	00:00:01	ora_s000_ORCL
oracle	6452	1	0	Feb08	?	00:01:39	ora_rvwr_ORCL
oracle	6468	1	0	Feb08	?	00:00:42	ora_arc0_ORCL
oracle	6470	1	0	Feb08	?	00:00:43	ora_arc1_ORCL
oracle	6472	1	0	Feb08	?	00:00:49	ora_arc2_ORCL
oracle	6474	1	0	Feb08	?	00:00:45	ora_arc3_ORCL
oracle	6476	1	0	Feb08	?	00:00:06	ora_smco_ORCL
oracle	6478	1	0	Feb08	?	00:00:51	ora_fbda_ORCL
oracle	6480	1	0	Feb08	?	00:00:04	ora_qmnc_ORCL
oracle	6510	1	0	Feb08	?	00:00:07	ora_q000_ORCL
oracle	6512	1	0	Feb08	?	00:00:13	ora_q001_ORCL
oracle	8969	1	0	Feb08	?	00:01:54	ora_cjq0_ORCL
oracle	20951	1	0	16:15	?	00:00:00	ora_w000_ORCL

Typical Sybase ASE Processes on Linux:

```

F S UID          PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S sybase    28414      1  0  80   0 -  5893 -          09:45 pts/3    00:00:00 /opt/sybase/ASE-15_0/bin/histserver -D/opt/sybase -SASE155_HS -Usa -l/opt/sybase/ASE-15_0/install/ASE155_HS.log -nl5
0 S sybase    27231      1  0  80   0 - 216722 futex_ 09:24 pts/2    00:00:01 /opt/sybase/ASE-15_0/bin/monserver -SASE155 -Usa -P -l/opt/sybase/ASE-15_0/install/ASE155_MS.log -L/opt/sybase/ASE-15_0/ASE155_MS.cfg -m/opt/sybase/ASE-15_0 -MASE155_MS
0 S sybase     8173      1  2  80   0 - 226520 -        Jun18 ?          00:48:06 /opt/sybase/ASE-15_0/bin/dataserver -d/opt/sybase/data/master.dat -e/opt/sybase/ASE-15_0/install/ASE155.log -c/opt/sybase/ASE-15_0/ASE155.cfg -M/opt/sybase/ASE-15_0 -sASE155
0 S sybase     4011      1  0  80   0 -  5653 -        Jun17 ?          00:00:00 /opt/sybase/ASE-15_0/bin/backupserver -e/opt/sybase/ASE-15_0/install/ASE155_BS.log -N25 -C20 -M/opt/sybase/ASE-15_0/bin/sybmultipbuf -SASE155_BS
sybase    28528 28166   0 09:46 pts/3    00:00:00 /bin/sh /opt/sybase/UAF-2_5/bin/uafstartup.sh
sybase    28555 28528   5 09:46 pts/3    00:00:06 /opt/sybase/shared/JRE-6_0_17_64BIT/bin/java -Djava.library.path=/opt/sybase/UAF-2_5/nodes/sybmain/rtlib -Dcom.sybase.ua.log.level=WARN -Djava.security.policy=/opt/sybase/UAF-2_5/conf/java.policy -Dcom.sybase.home=/opt/sybase -Dcom.sybase.ua.toplevel=/opt/sybase/UAF-2_5 -Dcom.sybase.ua.home=/opt/sybase/UAF-2_5/nodes/sybmain -classpath /opt/sybase/UAF-2_5/nodes/sybmain/server/lib/uaf-startup.jar com.sybase.ua.startup.Startup
sybase     8210 8173   0 Jun18 ?          00:00:01 /opt/sybase/ASE-15_0/bin/jsagent -p 4900 -n
sybmain -t 4 -l 0 -v -e /opt/sybase/ASE-15_0/install/ASE155_JSAGENT.log -i /opt/sybase > /dev/null

```

5.1.1 DATABASES AND INSTANCES

In Oracle, the main system components are defined in databases and instances, which sometimes are used inappropriately and cause confusion. For this guide we will use the official Oracle terminology:

- **Database:** A collection of physical operating system files or disk. When using Oracle 10g or 11g and the Automatic Storage Management (ASM), this might not appear as individual separate files in the operating system, but the definition for a database stays the same.
- **Instance:** A set of Oracle background processes/threads and a shared memory area, which uses memory that is shared across those processes/threads running on a single computer. A database instance can exist without any disk storage. This understanding helps to draw the line between a database and an instance in Oracle.

An instance is a set of processes/threads that operate on a database, a set of operating system files. At any time, an instance can only have one database associated with it. On the other hand a database can be associated to only one instance, but in an Oracle Real Cluster Application (RAC) environment a database is associated to multiple instances on multiple computers.



Applying a one-to-one translation from Oracle to Sybase ASE, an instance would be the `dataserver` process in Sybase ASE, which controls all the operations of a Sybase ASE server and a database would be the entire collection of system and user databases in Sybase ASE. Unlike Oracle, where an instance can start independently without opening a database, the Sybase ASE `dataserver` process will start the server and open every database. There are methods to tell the process to skip opening a specific database for recovery purposes.

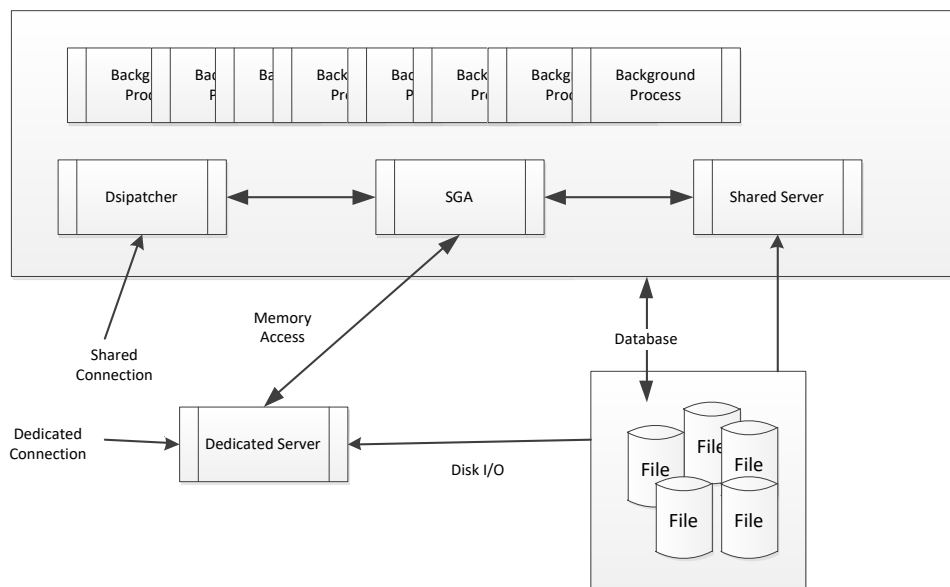
To complete the functions of an Oracle instance, the next Sybase process would be the Sybase Unified Agent. The Unified Agent is a key element in managing Sybase's cluster, called Sybase ASE Cluster Edition. This is Sybase's counterpart to Oracle's RAC system. Both systems are based on the shared everything cluster principle.

A typical startup of an Oracle server consists of launching the Oracle instance first, then mounting a database and lastly opening the database. Sybase ASE also has a function to mount and unmounts databases, but it is used in a backup and restore scenario to easily transfer database from one server to another.

5.1.2 CONNECTING TO ORACLE

When an application or other client tool connects to Oracle, a new process for this database user will be created. This is also called a dedicated server connection. These processes communicate directly with the database files and the memory. However, if the system has 5,000 active users this will lead into 5,000 separate processes or threads and could overwhelm the operating system. That's why Oracle introduced the Shared Server concept, where user processes share common memory and disk access constructs. This will save processes/threads, but it also introduces a new process called the dispatcher. Developers do not have to worry about connecting through a dedicated or shared server; this is purely a DBA management decision.

Oracle Connection Overview:



On Sybase ASE all user connections are shared and there is no need for a DBA to make a decision which way the server has to handle user connections. The only concern for a Sybase DBA is to limit the maximum user connections to a reasonable size to manage the memory overhead associated with user connections.

5.2 MEMORY STRUCTURE

Oracle makes clear distinctions between system, process and user memory.

Oracle maintains three major memory structures:

- **System Global Area (SGA):** This is a large, shared memory segment that virtually all Oracle processes will access at one point or another.
- **Process Global Area (PGA):** This is memory that is private to a single process or thread, and is not accessible from other processes/threads.
- **User Global Area (UGA):** This is memory associated with your session. It will be found either in the SGA or the PGA depending on whether you are connected to the database using shared server (then it will be in the SGA), or dedicated server (it will be in the PGA, the process memory).

The Sybase ASE user connection memory pool is probably the closest to Oracle's UGA. The procedure cache could be compared to the PGA and everything else would be compared to the SGA.

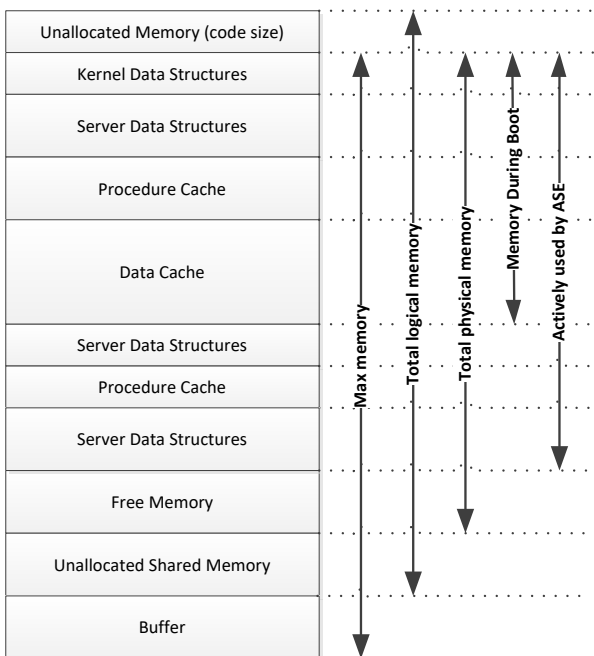


All memory allocation settings in Sybase ASE are dynamically, except for the max memory and changes do not require a reboot. This means that changes in data cache, user connection memory usage, procedure cache allocations and other memory constructs can be changed while the Sybase ASE server is running and no connections or transactions will be lost or disconnected. Even the physical hard memory limit set in the Sybase ASE kernel by the "max memory" parameter is subject to reevaluation when the system requires more memory at startup and the physical hardware is available.

5.2.1 SYBASE ASE ALLOCATES MEMORY AS FOLLOWS:

The total memory allocated during system start-up is the sum of memory required for all the configuration needs of Adaptive Server. This value can be obtained from the read-only configuration parameter total logical memory. This value is calculated by Adaptive Server. The configuration parameter max memory must be greater than or equal to total logical memory. max memory indicates the amount of memory you will allow for Adaptive Server needs.

During server start-up, by default, Adaptive Server allocates memory based on the value of total logical memory. However, if the configuration parameter allocate max shared memory has been set, then the memory allocated will be based on the value of max memory. The configuration parameter allocate max shared memory enables a system administrator to allocate the maximum memory that is allowed to be used by Adaptive Server, during server start-up.



The key points for memory configuration in Sybase ASE are:

- The system administrator should determine the size of shared memory available to Sybase ASE and set max memory to this value.
- The configuration parameter allocate max shared memory can be turned on during start-up and runtime to allocate all the shared memory up to max memory with the least number of shared memory segments. A large number of shared memory segments has the disadvantage of some

performance degradation on certain platforms. Check your operating system documentation to determine the optimal number of shared memory segments. Once a shared memory segment is allocated, it cannot be released until the server is restarted.

- The difference between max memory and total logical memory is additional memory available for the procedure and statement caches, data caches, or for other configuration parameters.

The amount of memory to be allocated by Sybase ASE during start-up, is determined by either total logical memory or max memory. If this value too high:

- Sybase ASE may not start if the physical resources on your machine are not sufficient.
- If it does start, the operating system page fault rates may rise significantly and the operating system may need to be reconfigured to compensate.
- Handling wider character literals requires Sybase ASE to allocate memory for string user data. Also, rather than statically allocating buffers of the maximum possible size, Sybase ASE allocates memory dynamically. That is, it allocates memory for local buffers as it needs it, always allocating the maximum size for these buffers, even if large buffers are unnecessary. These memory management requests may cause Sybase ASE to have a marginal loss in performance when handling wide-character data.
- Memory that is allocated dynamically slightly degrades the server's performance.
- When Sybase ASE uses larger logical page sizes, all disk I/Os are performed in terms of the larger logical page sizes. For example, if Sybase ASE uses an 8K logical page size, it retrieves data from the disk in 8K blocks. This should result in an increased I/O throughput, although the amount of throughput is eventually limited by the controller's I/O bandwidth.



Unlike Oracle, Sybase ASE does not offer an automatic memory management, because Sybase ASE dynamically allocates its memory. This causes a marginal performance decrease if wide-character data is being stored. Although Oracle offers the automatic memory management, in rapidly changing workload environments, this could lead to unintended problems. In general, automatic memory management in Oracle is more the exception than the rule.

5.3 PROCESSES

On UNIX systems Oracle's architecture is a multi-process system, whereas in Windows platforms it is a multi-threaded single process. Sybase ASE on the other hand is a multi-threaded, single process on any platform that Sybase ASE supports.

5.3.1 ORACLE PROCESS MANAGEMENT

Oracle maintains three broad classes of processes:

- **Server processes:** These perform work based on a client's request. The dedicated and shared servers are part of this process class.
- **Background processes:** These are the processes that start up with the database and perform various maintenance tasks, such as writing blocks to the disk, maintaining the online redo log, cleaning up aborted processes, and so on.
- **Slave processes:** These are similar to background processes, but they are processes that perform extra work on behalf of either the background or a server process.

Oracle Background Process List:

- **PMON** - Process Monitor
- **SMON** - System Monitor
- **RECO** - Distributed database recovery
- **CKPT** - Checkpoint process
- **DBWn** - Database block writer
- **LGWR** - Log writer
- **ARCn** - Archive process
- **MMON** - Manageability Monitor
- **MMNL** - Manageability Monitor Light
- **MMAN** - Memory Manager
- **CJQ0** - Job Queue Coordination
- **J000-J999** - Job Queue Process
- **FMON** - File Mapping Monitor

- **RVWR** - Recovery Writer
- **CTWR** - Change Tracking Writer
- **QMNC** - Queue Monitor Coordinator
- **BSP** - Block server process
- **LMON** - Lock monitor process
- **LMD** - Lock manager daemon
- **LCKn** - Lock process
- **Dnnn** - Dispatcher process
- **Snnn** - Shared Server process
- **PSP0** - Oracle process spawner
- **SHAD** - Oracle shadow process
- **q000 - q???** - Streams Advanced Queuing process
- **ACMS** - Atomic Control file to Memory Service
- **DBRM** - Database Resource Manager
- **DIA0** - Diagnosability Process 0
- **DIAG** - Diagnosability
- **EMNC** - Event Monitor Coordinator
- **FBDA** - Flashback Data Archiver process
- **SMCO** - Space Management Coordinator
- **VKTM** - Virtual Keeper of Time

5.3.2 SYBASE PROCESS MANAGEMENT

Sybase ASE starts a single process called `dataserver` to manage all of its database tasks as described in the broad process classes for Oracle.



In addition to this multi-threaded, single process, Sybase ASE utilizes the support from some additional processes and services that need to be started separately and are not part of the initial Sybase ASE server startup process.

These processes and services are:

- **Sybase Unified Agent:** This process provides runtime services to manage, monitor, and control distributed Sybase resources.
- **Sybase Backup Server:** This is an independent process that manages all backup processes performed on a Sybase database. With this server, backups can be performed locally as well as remotely.
- **Sybase XP Server:** This process allows operating system calls from the within the Sybase database server. Command shell execution and email delivery are amongst the most popular functions. This API calls can issued from any SQL statement construct and stored procedures as well.
- **Sybase Job Scheduler:** This process is providing the ability to define and schedule database administration tasks. With Job Scheduler, jobs that normally require interaction from a database administrator can be scheduled to run unattended at the appropriate times, freeing the database administrator to attend to other issues.
- **Sybase Monitor Server:** This is a process that provides a way to monitor Sybase ASE performance in real time or in a historical data-gathering mode. System administrators can use this information to identify potential resource bottlenecks, to research current problems, and to tune for better performance.
- **Sybase Historical Server:** This is the data archiving and gathering extension to the Sybase Monitor Server.
- **Sybase Web Services:** This is a process that enables client applications to access SQL using SOAP, enables the Sybase kernel to connect to other web services and provides user-defined web services to SOAP or a browser.

5.4 CONCURRENCY AND MULTI-VERSIONING

This is probably the most profound difference between Oracle and Sybase ASE. In an Oracle environment, deadlocks are a rare situation and the reason behind is Oracle's multi-versioning that provides non-blocking reads. On the other hand Sybase ASE places a shared lock for every read it performs. Although these shared locks are quickly released, while shared locks a place no exclusive locks can be obtained for the same data element (page or data row). Exclusive locks are necessary for any insert, update or delete operation. Hence reads in Sybase ASE are blocking, for a very short period of time, at least with the default isolation level 1. This is an important fact that Oracle developers need to embrace when migrating to Sybase ASE or developing new Sybase ASE applications.



Many Oracle DBA and developer heard about the infamous "deadlock on read" situation. Quite frankly this is mostly unheard to Sybase ASE developers, and the reason lies in the multi-versioning and non-blocking reads Oracle provides. Sybase ASE does not provide multi versioning of reads. The only way to provide the same behavior in Sybase ASE as the multi-versioning in Oracle is to force the ANSI compatible serializable isolation level or isolation level 3. Sybase ASE implements isolation level 3 with the holdlock parameter to every select statement. These locks will stay in place until the end of the transaction. While these locks are in place no DML operations can occur on these data elements. In a deadlock situation, Sybase ASE will declare a deadlock victim to solve the situation. That's when you see a SELECT statement declared as a deadlock victim, hence "deadlock on read". Because the underlying problem is a blocking situation that cannot be resolved until the transaction is completed, you can use a Sybase ASE server setting called "deadlock checking period" that determines the frequency of deadlock checks. Increasing this value might be a solution to a deadlock situation during a database migration. It will delay the check interval and this might just be long enough for the transactions to finish and locks to be released. The real solution is to rewrite the application to take full advantage of isolation level 1 and "deadlock on read" will never happen again.

Sybase ASE maintains the following locking schemes:

- Allpages locking, which locks data pages and index pages
- Datapages locking, which locks only data pages
- Datarows locking, which locks only data rows

5.4.1 ANSI ISOLATION LEVELS

The ANSI SQL standard defines four levels of isolation for transactions. For each isolation level there are 3 “phenomena” that describe which read actions are permitted or not. Both Sybase ASE and Oracle operate on the same isolation rules, but use different names for it.

Oracle	Isolation Level	Sybase ASE	Dirty Read	Nonrepeatable Read	Phantom Read
READ UNCOMMITTED	0		Permitted	Permitted	Permitted
READ COMMITTED (default)	1 (default)			Permitted	Permitted
REPEATABLE READ	2				Permitted
SERIALIZABLE	3				

- **Level 0 / Read Uncommitted:** In general this allows transactions to read uncommitted data from other transactions, hence “Dirty Reads”.
 - Sybase ASE maintains so called read scans that do not acquire any read locks so they do not block other transactions from writing to the same data, and vice versa. However, data modification statements (like update) still acquire read locks for their scans, because they must maintain the database integrity by ensuring that the correct data has been read before modifying it. Because read scans do not acquire any read locks, it is possible that the result set may change while the scan is in progress. If the scan position is lost due to changes in the underlying table, a unique index is required to restart the scan. In the absence of a unique index, the scan may be aborted.
 - Oracle is using multi-versioning to grant non-blocking read, therefore dirty reads in Oracle are much more harmful and can result in foreign key violations and unique constraint violations. Permitting “Dirty Reads” in Oracle are not necessary, but Oracle supports the spirit of “Dirty Reads” if necessary.
 -
- **Level 1 / Read Committed:** This prevents “Dirty Reads”, but does not prevent “Repeatable Read”. With this isolation level, it is possible for data modification statements to change the value of the data you just read before you complete your read transaction.

- However, in Sybase ASE applications it doesn't enforce data integrity. Other transactions can change the value if your transaction reads the data before the other transaction commits the changes. This is the default level for Sybase ASE.
- This is Oracle's default as well. The difference is that Oracle performs non-blocking reads to maintain read consistency.
- **Level 2 / Repeatable Read:** This prevents nonrepeatable reads. Such reads occur when one transaction reads a row and a second transaction modifies that row. If the second transaction commits its change, subsequent reads by the first transaction yield different results than the original read.
 - A transaction performing repeatable reads locks all rows or pages read during the transaction. After one query in the transaction has read rows, no other transaction can update or delete the rows until the repeatable-reads transaction completes. Transaction isolation level 2 is supported only in data-only-locked tables. If you use transaction isolation level 2 (repeatable reads) on allpages-locked tables, isolation level 3 (serializable reads) is also enforced. This is the extension of shared read locks to prevent cross transaction data modification that could change the read consistency.
 - Oracle handles this isolation level differently. It actually introduces a read lock. Unlike Sybase ASE, this read lock is never shared and as a result you will find rare Oracle read deadlocks with this isolation level. This is a rarely used isolation level in Oracle.
- **Level 3 / Serializable:** This ensures that data read by one transaction is valid until the end of that transaction, preventing phantom rows. Phantom rows are rows that have been added to the result set between the first read and a subsequent read. This is different from nonrepeatable reads in that data you already read has not been modified, but additional data has been added.
 - Sybase ASE uses range locks, infinity key locks, and next-key locks to protect against phantoms on data-only-locked tables. Allpages-locked tables protect against phantoms by holding locks on the index pages for the serializable read transaction. Sybase ASE supports this isolation level either by adding a `holdlock` attribute to each

select statement executed or setting it at the session level. This will prevent DML operations on the locked data as well as obtaining a holdlock on the same data for the duration of this transaction. Normal read operations that do not require a holdlock will still be able to access the data.

- Oracle takes a different approach to achieve serializable isolation. Instead of creating read locks on the data, it reads subsequent data from the rollback segment. This is a very optimistic approach and requires that the data stays long enough in the rollback segment to be accessible.

In addition to the 4 isolation levels ANSI SQL requires, Oracle also introduced a fifth level called "Read Only". This is a different flavor of the level 3/ Serializable isolation level.

5.5 SCHEMA MIGRATION

When comparing schemas between Oracle and Sybase ASE it is important to understand what a schema in Oracle means. In Oracle, a schema is equivalent to a user login. There might be some objections to that, but it is the general assumed practice. A schema is a collection of objects that adhere to a common set of security rules. A user login can exist without a schema, but a schema always is a user login.

Without creating application objects, there are already 2 schemas in any Oracle database present, sys and system. They contain tables, views, indexes, users, constraints, stored procedures, triggers, and other database-specific objects.

5.5.1 SCHEMA OBJECT SIMILARITIES

The following table outlines the schema object similarities.

Oracle	Sybase ASE
Database	Database
Schema	Database and objects owned by the same user.
Tablespace	Database
User	User
Role	Role
Table	Table
Temporary tables	Temporary tables
Cluster	Clustered index on datapage locking tables
Column-level check constraint	Column-level check constraint
Column default	Column default
Unique key	Unique key or identity property for a column
Primary key	Primary key
Foreign key	Foreign key
Index	Non-unique index
PL/SQL Procedure	Transact-SQL (T-SQL) stored procedure
PL/SQL Function	T-SQL stored function

Oracle	Sybase ASE
Packages	N/A
AFTER triggers	Triggers
BEFORE triggers	Defined rules at the table definition level for each column the BEFORE trigger contains an action
Triggers for each row	N/A
Synonyms	Similar functionalities can be emulated via views for table and view synonyms
Sequences	Identity property for a column or dedicated key value table
Snapshot	N/A
View	View
Materialized Views	Oracle uses materialized views for one way replication via dblink connections and updateable materialized views for two-way replication. Sybase Replication Server will handle these replication needs outside Sybase ASE.

5.5.2 SCHEMA OBJECT NAMES

Keep in mind that the reserved words differ between Oracle and Sybase ASE. Before attempting a schema migration, all Oracle objects need to be checked against the Sybase ASE reserved words.

For a complete list of reserved words in Sybase ASE, see *"Adaptive Server Enterprise->Reference Manual: Building Blocks->Reserved Words"*

5.5.3 TABLE DESIGN CONSIDERATIONS

When migrating a schema from Oracle to Sybase ASE there will be situations where special methods need to be applied to complete the migration and ensure the data is correctly represented. Special considerations have to be placed on data types and different types of constraints.

5.5.3.1 DATA TYPES

There are 3 data types that need special attention when migrating an Oracle schema to Sybase ASE.

5.5.3.1.1 DATETIME DATA TYPES

Oracle's default date data type is `DATE`. Its granularity is up to the second. Loading `DATE` data into Sybase ASE's `datetime` data type is not a problem. Sybase ASE maintains a 1/300th of a second in the `datetime` data type. If you want to map the `TIMESTAMP` data type from Oracle to Sybase ASE, the `bigdatetime` would be the next best data type. The `TIMESTAMP` data type has a granularity of 1/100000000th of a second. Sybase's `bigdatetime` data type stores the microseconds since midnight for the time part.

5.5.3.1.2 IMAGE AND TEXT DATA TYPES

Oracle stores large binary objects in the `BLOB` data type and large character objects in the `CLOB` data type. Both data types can store up to 128TB (4GB * database block size) of data, as of Oracle 11g. Even the `VARCHAR2` data type can store a maximum of 4000 bytes.

On Sybase ASE the maximum length of the `varchar` data type is defined by the page size of the database. Assuming that the Sybase ASE server was installed with the default page size of 4kb, the maximum length of the `varchar` data type would be 4kb, minus some overhead bytes. If the server was created with the smaller 2kb page size, the only alternative would be mapping `VARCHAR2` data types from Oracle to `TEXT` data types in Sybase ASE.

Migrating data from Oracle's `BLOB` data type will be mapped to `IMAGE` data types in Sybase ASE and `CLOB` to `TEXT` data types. The max size for `IMAGE` and `TEXT` data types in Sybase ASE are 4GB. If there are larger storage requirements than Sybase ASE can handle, Sybase IQ provides to option of matching Oracle's `BLOB` and `CLOB` storage capacity.

5.5.3.1.3 USER DEFINED DATA TYPES

The mapping of user-defined data types is very difficult and requires extensive manual intervention to determine the base data type. For Oracle, this is an add-on option to the database and not widely used. The key to user-defined data type migration is to fully understand the underlying base data type. This could be nested in several iterations.

5.5.3.2 CONSTRAINTS

- Entity Integrity Constraints are defined by a primary key. Primary keys can be incorporated in a CREATE TABLE or ALTER TABLE statement, in both Sybase ASE and Oracle. Migration is straight forward.
- Referential Integrity Constraints are defined by a foreign key constraint. Both Oracle and Sybase ASE offer foreign key constraints in the CREATE TABLE and ALTER TABLE statement.
- Unique Key Constraints can be defined in the CREATE TABLE and ALTER TABLE statement on both Oracle and Sybase ASE. Unique keys map one-to-one from Oracle to Sybase ASE.
- Check Constraints are defined in Oracle on a TABLE level only, whereas Sybase ASE allows check constraints on the COLUMN level. Oracle's table level check constraints map one-to-one with Sybase ASE's table level check constraints.

5.6 DATA TYPES

This section provides detailed description of differences in data types and how Oracle data types should be mapped to Sybase ASE data types.

Oracle	Description	Sybase ASE	Comments
NUMBER(x)	Oracle number data types with 0 decimal can be converted into equivalent Sybase ASE data types.	BIGINT	If length of the NUMBER data type is greater than 15.
		FLOAT	If length of the NUMBER data type is between 11 and 15
		INTEGER	If the length of the NUMBER is between 6 and 10
		SMALLINT	If the length of the NUMBER is between 4 and 5
		TINYINT	If the length of the number is between 2 and 3 and the data value is <= 255
		BIT	If the length of the NUMBER is 1
NUMBER(x,y)	Oracle number data type with greater than 0 decimal can be converted into these Sybase ASE data types	DOUBLE	If the precision of the NUMBER field is > 15
		FLOAT	If the precision of the NUMBER field is <= 15
NUMBER(x,y)	Alternatively to the mapping path above, these Sybase ASE data types can be used.	NUMERIC(x,y)	This translates the Oracle NUMBER data type one-to-one.
		DECIMAL(x,y)	
		MONEY	If the Oracle data type is NUMBER(x,2) and the data value represents currency.
FLOAT	Max FLOAT precision in Oracle is approx. 38.	DOUBLE	If the precision of the NUMBER field is > 15

Oracle	Description	Sybase ASE	Comments
		FLOAT	If the precision of the NUMBER field is <= 15
CHAR(x)	Max CHAR size in Oracle is 2000 bytes	CHAR(x)	If database page size is 4kb or greater. If the page size is 2kb and the CHAR(x) size is less than 1953 bytes.
		TEXT	If database page size is 2kb and the CHAR(x) size is greater than 1953 bytes.
VARCHAR2(x)	Max VARCHAR2 size in Oracle is 4000 bytes	VARCHAR(x)	If database page size is 8kb or greater. If the page size is 4kb and the CHAR(x) size is less than 4001 bytes. If the page size is 2kb and the CHAR(x) size is less than 1953 bytes.
		TEXT	If none of the conditions above matches.
DATE	The DATE / Time precision in Oracle is up to one second.	DATETIME	Sybase ASE's datetime precision is up to 1/300 th of a second.
TIMESTAMP	Oracle's TIMESTAMP has a precision of 1/100000000 th of a second.	BIGDATETIME	This Sybase ASE date type is almost able to match Oracle's precision.
RAW(x)	The RAW data type in Oracle has a max precision of 2000 bytes	BINARY(x)	If database page size is 4kb or greater. If the page size is 2kb and the RAW(x) size is less than 1953 bytes.
		IMAGE	If database page size is 2kb and the RAW(x) size is greater than 1953 bytes.
ROWID	This is a pseudo column in Oracle and does not represent a true data type. It is a hexadecimal code of 10 bytes in length.	CHAR(10)	This might be necessary for cross reference purposes.
CLOB	Oracle's max storage capacity for CLOB is 128TB.	TEXT	Sybase ASE can hold a maximum of 4GB in this data type.

Oracle	Description	Sybase ASE	Comments
BLOB	Oracle's max storage capacity for BLOB is 128TB.	IMAGE	Sybase ASE can hold a maximum of 4GB in this data type.
CHAR(1)	If this is a packed bit field maintained by a PL/SQL function set / unset / retrieve / query on them.	BIT	

These are the most common data types to be migrated from Oracle to Sybase ASE. As a word of caution, only convert data types into either TEXT or IMAGE in Sybase ASE when absolutely necessary. There are restrictions on those data types that could impact the application performance.

Appendix C contains the complete list of data types.

5.7 DATA STORAGE CONCEPTS

The physical architecture is made up of files that contain the system (or catalog) and application data. As with Oracle, Sybase ASE also has support for raw devices. The physical architecture is used to provide separation of data based on its type, such as metadata from user data; heap data from index data; user data from DBMS data (including transaction logs), and permanent data from temporary data.

The Oracle physical architecture is made up of data files, redo log files, and control files.

5.7.1 DATAFILES

The datafiles contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the datafiles allocated for a database. For ease of management, one or more datafiles can be grouped into logical tablespaces. Each database is logically divided into one or more tablespaces. A datafile can be associated with only one database.

5.7.2 CONTROL FILES

A control file contains entries that specify the physical structure of the database. It contains the following information:

- Database name
- Names and locations of datafiles and redo log files
- Time stamp of database creation

5.7.3 REDO LOG FILES

The set of redo log files is collectively known as the redo log for the database. A redo log is made up of redo entries.

The primary function of the redo log is to record all changes made to data. If a failure prevents modified data from being permanently written to the datafiles, then the changes can be obtained from the redo log, so work is never lost.



Oracle is not case sensitive. Whereas Sybase ASE is case sensitive by default. This can be changed, however, when creating the ASE database server. Keywords within Sybase will always be case insensitive. Below is an example of using case sensitivity.

Creating a table with then name TEST in Sybase ASE requires you to always use TEST in uppercase to access the object in the server.

Select * from TEST => Result sets are being displayed

Select * from test => Error message: object nor found

5.7.4 DATA STORAGE CONCEPT MAPPING

Oracle	Sybase ASE
<p>Data Files</p> <p>One or more data files are created for each tablespace to physically store the data of all logical structures in a tablespace. The combined size of the data files in a tablespace is the total storage capacity of the tablespace. The combined storage capacity of the tablespaces in the database is the total storage capacity of the database.</p>	<p>Database Devices</p> <p>A database device is mapped to a specific physical disk file or raw device. Database devices are shared amongst one or more databases. The database devices can be tuned to achieve the best performance throughput on the specific operating system.</p> <p>The database and logs are stored on database devices. Each database device must be initialized before being used for database storage. Initialization of the database device initializes the device for storage and registers the device with the server. After initialization the device can be:</p> <ul style="list-style-type: none"> • Allocated to the free space available to a database • Allocated to store specific user objects • Used to store the transaction log of a database • Labeled as default device to create and alter database objects. <p>The SP_HELPDEVICE system procedure displays all the devices that are registered with the server. Use the <code>sp_dropdevice <logical device name></code> command to drop the device. The system administrator (sa) should restart the server after dropping the device to release locks on the physical disk device or file.</p> <p>A device can be labeled as a default device so that the new database need not specify the device at the time of creation. Use the SP_DISKDEFAULT system procedure to label the device as a default device.</p>

Oracle	Sybase ASE
<p>Data Block</p> <p>One data block corresponds to a specific number of bytes, of physical database space, on the disk. The size of the data block can be specified when creating the database.</p>	<p>Page</p> <p>Many pages constitute a database device. Each page contains a certain number of bytes. The size of a page can be specified when installing the Sybase ASE server and creating the databases for the first time. This size cannot be change, unless the Sybase ASE server will be installed from scratch.</p>
<p>Extent:</p> <p>An extent is a specific number of contiguous data blocks, obtained in a single allocation.</p>	<p>Extent</p> <p>Eight pages make one extent. Space is allocated to all the databases in increments of one extent at a time. Sybase ASE avoids wasting extra space by filling up existing allocated extents in a target allocation page, even though these extents are assigned to other partitions. The net effect is that extents are allocated only when there are no free extents in the target allocation page.</p>
<p>Segments:</p> <p>A segment is a set of extents allocated for a certain logical structure. The extents of a segment may or may not be contiguous on disk, may or may not span the data files.</p>	<p>N/A</p>
<p>Tablespaces</p> <p>A database is divided into logical storage units called tablespaces. A tablespace is used to group related logical structures together. A database typically has one system tablespace and one or more user tablespaces.</p>	<p>Databases</p> <p>A Sybase ASE server consists of multiple databases. Several system databases make up the core and one or more user database are being added.</p>
<p>Tablespace Extent:</p> <p>An extent is a specific number of contiguous data blocks within the same tablespace.</p>	<p>Segments</p> <p>A segment is the name given to one or more database devices. One or more database devices make a database. Segment names are used in CREATE TABLE and CREATE INDEX constructs to place these objects on specific database devices. Segments can be extended to include additional devices as and when needed by using</p>
<p>Tablespace Segments:</p> <p>A segment is a set of extents allocated for a certain logical</p>	

Oracle	Sybase ASE
<p>database object. All the segments assigned to object must be in the same tablespace. The segment gets the extents allocated to them as and when needed.</p> <p>There are four different types of segments as follows:</p> <ul style="list-style-type: none"> • Data segment Each table has a data segment. All of the table's data is stored in the extents of its data segment. The tables in Oracle can be stored as clusters as well. A cluster is a group of two or more tables that are stored together. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment. • Index segment Each index has an index segment that stores all of its data. • Rollback segment One or more rollback segments are created by the DBA for a database to temporarily store "Undo" information. This is the information about all the transactions that are not yet committed. This information is used to generate read-consistent database information during database recovery to rollback uncommitted transactions for users. • Temporary segment Temporary segments are created by Oracle when a SQL statement needs a temporary work area to complete execution. When the statement finished execution, the extents in the temporary segment are returned to the system for future use. 	<p>the SP_EXTENDSEGMENT system procedure.</p> <p>The following segments are created along with the database:</p> <ul style="list-style-type: none"> • System segment Stores the system tables • Log segment Stores the transaction log • Default segment All other database objects are stored on this segment unless specified otherwise. <p>Segments are subsets of database devices.</p> <p>The data and index segment of an Oracle tablespace would be mapped to the default segment in Sybase ASE.</p> <p>Oracle tablespaces do not need system tables, therefore there is no mapping. Oracle keeps its system tables in a SYSTEM tablespace in the data and index segments.</p> <p>Sybase ASE maintains the committed and uncommitted transactions in the same transaction log. This is equivalent to the tablespace rollback segment and the redo log files.</p> <p>In Sybase ASE temporary work areas and temporary tables are maintained in the tempdb. Sybase ASE allows you to create and manage multiple temporary databases in addition to the system temporary database, tempdb. Multiple temporary databases reduce contention on system catalogs and logs in tempdb.</p>
<p>Redo Log Files: Each database has a set of two or more redo log files. All changes made to the database are recorded in the redo log. Redo log files are critical in protecting a database against failures. Oracle allows mirrored redo log files so that two or more copies of these files can be maintained. This</p>	<p>Log Devices: These are logical devices assigned to store the log. The database device to store the logs can be specified while creating the database.</p> <p>Sybase ASE allows mirroring log devices separately from</p>

Oracle	Sybase ASE
protects the redo log file against failure of the hardware the log file resides on.	data devices. This will protect the database from hardware failure. With mirrored log devices transactions can be restored up to the last committed transaction.
N/A	<p>Dump Devices:</p> <p>These are logical devices. A database dump is stored on these devices. The DUMP DATABASE command uses the dump device to dump the database.</p>
<p>Control Files:</p> <p>Each database has a control file. This file records the physical structure of the database. It contains the following information:</p> <ul style="list-style-type: none"> • Database name • Names and location of a database's data files and redo log files • Timestamp of database creation <p>It is possible to have mirrored control files. Each time an instance of an Oracle database is started, its control file is used to identify the database, the physical structure of the data, and the redo log files that must be opened for the database operation to proceed. The control file is also used for recovery if necessary.</p>	<p>Master Database</p> <p>The master database in Sybase ASE controls more than just the database information, but it is as vital to the server startup as the Oracle control files.</p> <p>The system tables involved on the master database to maintain similar information as the Oracle control files are:</p> <ul style="list-style-type: none"> • sysdevices • sysdatabases • sysusages • systhresholds • syssegments • syspartitions • sysindexes <p>Sybase ASE allows the master database to be mirrored as well to protect against hardware failures. Frequent backups of the master database are most important to protect against catastrophic failures. Including failed data manipulation by an administrator.</p>

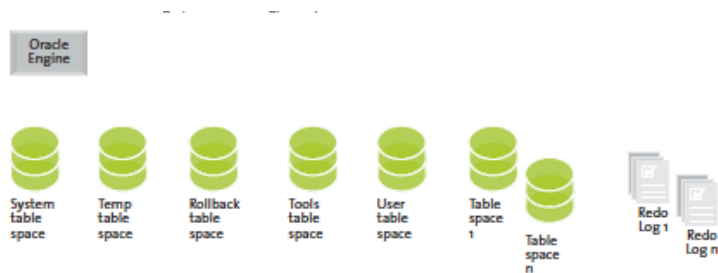


The conceptual differences in the storage structures do not affect the conversion process directly. However, the physical storage structures need to be in place before the conversion of the database begins.

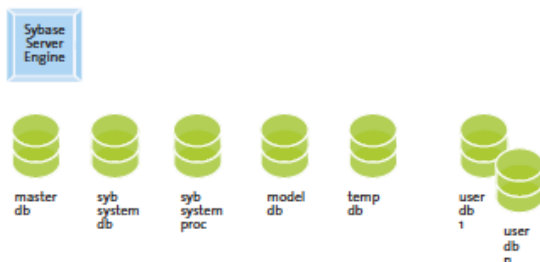
5.7.5 OVERVIEW OF THE MAIN COMPONENTS OF THE ORACLE DATABASE

The data consists of system and user data:

- System table space, contains information on system and user objects
- Temp table space, is the scratch area to manage transactions
- Tools table space, objects needed by tools.
- User table space, for a user's personal objects
- Rollback table space, to store undo information for managing transactions
- Two or more Redo logs (transaction log) is a special OS file



5.7.6 OVERVIEW OF THE MAIN COMPONENTS OF THE SYBASE ASE SERVER



The Sybase ASE database uses logical devices to store the physical data. Sybase logical devices are created on OS devices (which can be stripped over several disks). The database is created on one or more logical devices. One logical device can contain multiple databases. A database can consist of multiple segments each on their own logical device. Tables and indexes can be placed on different segments. The use of segments allows the developer to control where the data is placed in order to maximize I/O performance.

The Sybase ASE server consists of system databases, optional database and user databases. The system databases included on installation of Sybase ASE are:

Database Name	Description
master	Controls the operation of the Sybase ASE as a whole and stores information about the all users, users databases, devices, system table entries, server configuration
model	Template for all user databases. Each time a user enters the create database command, Sybase ASE makes a copy of the <i>model</i> database and extends the new database to the size specified by the create database command. Model database can be modified to customize the newly created user databases.
sybtempprocs	Storage of system wide stored procedures. When a user in any database executes a system stored procedure (that is, a procedure whose name begins with sp_), Sybase ASE first looks for that procedure in the user's current database. If there is no procedure there with that name, Sybase ASE looks for it in <i>sybtempprocs</i> . If there is no procedure in <i>sybtempprocs</i> by that name, Sybase ASE looks for the procedure in <i>master</i> .
tempdb	Temporary working storage and sort activity, used for creating temporary user tables. The space in <i>tempdb</i> is shared among all users of all databases on the server.

The optional databases can be installed optionally are:

Database Name	Description
sybsecurity	Contains the audit system for Sybase ASE
sybtempdb	Stores information about distributed transactions

Database Name	Description
sybmgmtadb	Stores jobs, schedules, scheduled jobs information, and data the internal Job Scheduler task needs for processing
<i>pubs2</i> and <i>pubs3</i> sample databases	Provided as a learning tool for Sybase ASE

User databases are defined by the developer, and they contain the application data. The developer may decide to put business data in different user databases.

Every database has a system table that contains the transactions for recovery purposes. In the transaction log both the before and after images are stored.

The Sybase ASE server is case sensitive by default, if needed this can be changed when creating the Sybase ASE instance.

System tables exist in both system and user databases, and they contain information about the different objects:

- tables
- views
- indexes (including Primary Key and Foreign Key)
- constraints (defaults, rules)
- triggers
- stored procedures
- stored functions
- users

The master database must be available to access all other databases. It contains system tables that are only available in the master database. The objects contained in these system tables are for example:

- devices
- databases
- logins
- remote logins

- processes
- storage space
- Active locks
- Server Roles
- engines
- configuration

In Oracle, the smallest unit of storage is the data block. All data in the database is retrieved and manipulated in terms of blocks. DBA can choose the default page size at the time of database creation to be any of 2K, 4K, 8K, 16K or 32K.

The basic unit of storage for Sybase ASE is a page. Page sizes can be 2K, 4K, 8K to 16K. The server's page size is established when you first build the source. Once the server is built the value cannot be changed. These types of pages store database objects:

- **Data pages** – store the data rows for a table.
- **Index pages** – store the index rows for all levels of an index.
- **Large object (LOB) pages** – store the data for text and image columns, and for Java off-row columns.

All pages have a header that stores information such as the object ID that the page belongs to and other information used to manage space on the page. The rest of the page is available to store data and index rows.

In Sybase ASE, the developer can choose the locking scheme of the database user tables. The default-locking scheme is allpage locking, and this can be changed server wide. It is also possible to set the locking scheme for just a table when creating the table. In allpage locking the index pages and datapages are locked. All rows on the page that is locked are inaccessible. Data-only-locking (DOL) only locks the data pages. There are two flavors, DOL datapage locking and row locking. Datapage locking locks the datapage, and all rows on the locked page are inaccessible. Row locking just locks a row on a datapage, making all other rows on the page accessible.

Sybase ASE may have one clustered index per table. For the tables with allpage locking scheme this means that the leaf page of the data is in the order of the clustered index. For tables with the data-only-locking scheme, the data is put in the clustered order where possible.

5.8 DATA MANIPULATION LANGUAGE

5.8.1 SQL LANGUAGE

There may be significant migration issues with the SQL Language since Oracle has implemented extensions to the SQL-92 SQL Languages standard. If the product needs to be deployed on multiple RDBMS's the developer should restrict the SQL used to entry-level SQL-92 standard. Otherwise the developer will need conditional code for different RDBMS's.

Appendix D describes the SQL language differences that need to be addressed in the migration process. It indicates whether the change can be handled automatically or has to be done manually. This is an in-depth side by side comparison of language changes that need to occur for an Oracle application to run on Sybase ASE, including some code examples.

5.8.2 CONCEPTUAL SQL LANGUAGE DIFFERENCES

Most queries can be migrated with little or no manual rewriting. The following comparison between Oracle and Sybase ASE syntax outlines what to look out for and to identify potential issues. The biggest challenge is to migrate the more complex analytical queries. A typical OLTP application issues short and precise queries against a small set of tables at one time.

5.8.2.1 CONNECTING TO THE DATABASE

Oracle	Sybase ASE
Syntax:	Syntax:
CONNECT user_name/password SET ROLE	USE database_name
	Description:
	The default database is assigned to each user. This database is made current when the user logs on to the server. A user executes the USE DATABASE_NAME command to switch to another database.



The concept of connecting to a database is conceptually different in Oracle and Sybase ASE. An Oracle Server controls only one database and to switch schema access the user executes the SET ROLE command to change roles or re-issues a CONNECT command using a different user_name or schema name, as we learned earlier. A Sybase ASE user can log on to the server and switch to another database residing on the server, provided the user has privileges to access the database.

5.8.2.2 SELECT STATEMENT

The statement in the table below retrieves rows from one or more tables and views.

Oracle	Sybase ASE
<p>Example 1:</p> <pre> SELECT DECODE(T1.C1, 'CLNN', T2.C2, T3.C3) as P_ID, T4.ID as ID FROM T4, T1, T2, T3 WHERE T4.ID = T1.ID(+) and T1.ID(+) = 'L3' and T4.ID = T2.ID(+) and T4.ID = T3.ID(+) and T3.TYPE='C' and T3.STAT = '00' </pre>	<p>Example 1:</p> <pre> SELECT case T1.C1 when 'CLNN' then T2.C2 else T3.C3 end as P_ID, T4.ID as ID into tempdb..query13 FROM T1 left outer join T2 on T1.ID = T2.ID and T1.ID = 'L3' left outer join T3 on T1.ID = T3.ID left outer join T4 on T1.ID = T4.ID where T3.TYPE='C' and T3.STAT = '00' </pre>
<p>Example 2:</p> <pre> SELECT T1.EFF_DATE, NVL(T1.END_DATE, 0) as END_DATE, T1.ID FROM T1 WHERE T1.STAT='00' AND (T1.ID, T1.TIMESTAMP) IN (SELECT T1.ID, MAX(T1.TIMESTAMP) as TIMESTAMP FROM T1 WHERE T1.STAT='00' GROUP BY T1.ID) </pre>	<p>Example 2:</p> <pre> select T1.EFF_DATE, isnull(T1.END_DATE,0) as END_DATE, T1.ID from (select EFF_DATE, END_DATE, ID, TIMESTAMP from T1 where STAT='00') as T1, (select ID, max(TIMESTAMP) as TIMESTAMP from T1 where STAT='00' group by ID) as T2 where T1.ID = T2.ID </pre>

Oracle	Sybase ASE
	and T1.TIMESTAMP = T2.TIMESTAMP
Description:	Description:
DISTINCT eliminates the duplicate rows.	DISTINCT eliminates the duplicate rows.
The INSERT INTO <table> SELECT FROM.... construct allows you to insert the results of the SELECT statement into a table.	The INTO clause and the items that follow it in the command syntax are optional, because Sybase ASE allows SELECT statements without FROM clauses as can be seen in the following example:
COLUMN ALIAS is defined by putting the alias directly after the selected COLUMN.	SELECT getdate()
If you use TABLE ALIAS, the TABLE must always be referenced using the ALIAS.	SELECT...INTO allows you to insert the results of the SELECT statement into a table.
You can also retrieve data from SYNONYMS. EXPRESSION could be a column name, a literal, a mathematical computation, a function, several functions combined, or one of several PSEUDO-COLUMNS.	SELECT_LIST can contain a SELECT statement in the place of a column specification as follows:
If a GROUP BY clause is used, all non-aggregate select columns must be in a GROUP BY clause.	SELECT d.empno, d.deptname, empname = (SELECT ename FROM emp WHERE enum = d.empno) FROM dept d WHERE deptid = 10
The FOR UPDATE clause locks the rows selected by the query. Other users cannot lock these rows until you end the transaction.	The preceding example also shows the format for the column alias.
Flashback-Query Clause: This allows selecting data of a specific timestamp from the flashback area.	ALIAS = selected_column
	COMPUTE attaches computed values at the end of the query. These are called row aggregates. If a GROUP BY clause is used, all non-aggregate select columns are needed.
	Flashback-Queries are not available in Sybase ASE.
	Pivot clauses, hierarchical clauses, analytical cube clauses and model clauses are not supported by Sybase ASE, but are supported by Sybase IQ.



In an Oracle database retrieving system information requires a SELECT statement that executes against a table with one row. To ensure that there is a table with one row at all times, Oracle created the DUAL table. Retrieving the system date via SQL looks like this in Oracle:

```
SELECT sysdate FROM dual;
```

Sybase ASE supports SELECT statements that do not have a FROM clause. The same query in Sybase ASE would look like this:

```
SELECT getdate()
```

If you don't want to rewrite the SELECT from DUAL statements for Sybase ASE, you have the option to create a dummy DUAL table in the Sybase ASE user databases with one row entry containing only one column. Be aware that Sybase ASE is case-sensitive and you might need to create at least 2 dummy tables, DUAL and dual. Depending on how disciplined the Oracle developers were.

5.8.2.2.1 DUAL TABLE DEFINITION

```
scott@oral> desc dual
Name                               Null?    Type
-----
DUMMY                               VARCHAR2(1)
```

```
scott@oral> select * from dual;
```

```
D
-
X
```

5.8.2.3 INSERT STATEMENT

Oracle	Sybase ASE
Example 1:	Example 1:
<pre>INSERT INTO state (state_abbrev) VALUES ('WA');</pre>	<pre>INSERT INTO state (state_abbrev) VALUES ('WA')</pre>
<pre>COMMIT;</pre>	<pre>GO</pre>
<pre>SELECT * FROM state;</pre>	<pre>SELECT * FROM state</pre>
	<pre>GO</pre>

/

Description:

INTO is required.

Inserts can only be done on single table views.

Description:

INTO and PLAN are optional.

Inserts are allowed in a view provided only one of the base tables is undergoing change.



The value supplied in the VALUES clause in either database may contain functions. The Oracle functions must be replaced with the equivalent Sybase ASE functions.

See Appendix D for a complete list of built-in function comparison.

5.8.2.4 UPDATE STATEMENT

Oracle

Example 1:

```
DECLARE
  v_salary emp.salary%TYPE;

  CURSOR c_emp IS
    SELECT *
    FROM emp
    WHERE deptid IN ('01')
    FOR UPDATE OF salary;

BEGIN
  -- Set up the cursor fetch loop.
  FOR v_empInfo IN c_emp LOOP
    -- Get salary increase based on rating
    SELECT sal_increase
    INTO v_salary
    FROM sal_increase_by_rating
    WHERE rating = ABS(v_empinfo.rating)
    -- Update the row we just retrieved from
the cursor.
    UPDATE emp
    SET salary = salary + v_salary
    WHERE CURRENT OF c_emp;
  END LOOP;

  -- Commit our work.
  COMMIT;
END;
```

Description:

A single subquery may be used to update a set of columns. This subquery must select the same number of columns (with compatible data types) as are used in the list of columns in the SET clause.

The CURRENT OF cursor clause causes the UPDATE statement to affect only the single row currently in the cursor as a result of the last FETCH. The cursor SELECT statement must have included in the FOR UPDATE clause.

Updates can only be done on single table views.

Sybase ASE

Example 1:

```
UPDATE emp
SET salary = salary + b.sal_increase
FROM sal_increase_by_rating b
WHERE deptid IN ('01')
AND ABS(rating) = b.rating
```

Description:

The FROM clause is used to get the data from one or more tables into the table that is being updated or to qualify the rows that are being updated.

Updates through multi-table views can modify only columns in one of the underlying tables.

5.8.2.5 DELETE STATEMENT

Oracle	Sybase ASE
Example 1: <pre>DELETE FROM SALES_HIST WHERE SALES_DATE <= '1999/12/31' /</pre>	Example 1: <pre>DELETE FROM SALES_HIST WHERE SALES_DATE <= '1999/12/31' GO</pre>
Descriptions: FROM is optional. ALIAS can be specified for the table name as a correlation name, which can be used in the condition. Deletes can only be performed through single table views	Descriptions: The first FROM in DELETE FROM is optional. The second FROM clause is a Sybase ASE extension that allows the user to make deletions based on the data in other tables. A sub query in the WHERE clause serves the same purpose. Deletes can only be performed through single table views.

5.8.2.6

5.8.2.7 LOCKING CONCEPTS AND DATA CONCURRENCY ISSUES



Locking serves as control mechanism for concurrency. Locking is necessary in a multi-user environment because more than one user at a time may be working with the same data. The locking concept in Sybase ASE is very different from Oracle's locking concept. This is due to the architecture differences and Oracle's versioning mechanism that allows non-blocking read access to the data.

5.8.2.7.1 LOCKING

Oracle	Sybase ASE
Oracle locking is fully automatic and does not require intervention by users. Oracle features the following categories of locks:	Sybase ASE locking is fully automatic and does not require intervention by users.
Data locks (DML locks) to protect data. The "table locks" lock the entire table and "row locks" lock individual rows.	There are differences in locking methods between allpages-locked tables and datarow-locked tables.
Dictionary locks (DDL locks) to protect the structure of objects.	Allpages-locked Tables:
Internal locks to protect internal structures, such as files.	Both data pages and index pages are being locked.
DML operations can acquire data locks at two different levels; one for specific rows and one for entire tables.	SELECT: Shared page lock INSERT: Exclusive page lock on data page and exclusive page lock on leaf-level index page DELETE: Update page locks followed by exclusive page locks on data pages and exclusive page locks on leaf-level index pages UPDATE: Update page lock in data page and exclusive page lock on data page
Row-level locks:	Datapages-locked Tables:
An exclusive lock is acquired for an individual row on behalf of a transaction when the row is modified by a DML statement. If a transaction obtains a row level lock, it also acquires a table (dictionary) lock for the corresponding table. This prevents conflicting DDL (DROP TABLE, ALTER TABLE) operations that would override data modifications in an on-going transaction.	Only data pages are being locked. No transactional locks are held on index pages.
Table-level data locks can be held in any of the following modes:	SELECT: Shared page lock on data pages INSERT: Exclusive page lock on data page DELETE: Update page locks followed by exclusive page locks on data pages UPDATE: Update page lock in data page
Row share table lock (RW):	Datarow-locked Tables:
This indicates that the transaction holding the lock on the table has locked rows in the table and intends to update	

Oracle

them. This prevents other transactions from obtaining exclusive write access to the same table by using the LOCK TABLE table IN EXCLUSIVE MODE statement. Apart from that, all the queries, inserts, deletes, and updates are allowed in that table.

Row exclusive table lock (RX):

This generally indicates that the transaction holding the lock has made one or more updates to the rows in the table. Queries, inserts, deletes, updates are allowed in that table.

Share lock (SL):

Share row exclusive lock(SRX)

Exclusive lock (X):

The dynamic performance table V\$LOCK keeps the information about locks.

Sybase ASE

SELECT: Shared row lock
INSERT: Exclusive row lock
DELETE: Update row locks followed by exclusive row locks on each affected row
UPDATE: Update row locks followed by exclusive row locks on each affected row

For non-update or read operations, a shared lock is applied. If a shared lock is applied to a table or a page, other transactions can also obtain a shared lock on that table or page. However, no transaction can obtain an exclusive lock. Therefore, Sybase ASE reads block the modifications to the data.

Extent locks:

Extent locks lock a group of eight database pages while they are being allocated or freed. These locks are held during a CREATE or DROP statement, or during an INSERT that requires new data or index pages.

A list of active locks for the current server can be seen with SP_LOCK system procedure.



Because Oracle's architecture allows for non-blocking reads it is immense important to check the application for a) excessive deadlocks after migrating the application and b) excessive "blocks" that have a negative impact on performance. Applying datarow-level locking to the tables will help mitigating these problems. This is one of the hot spots in an Oracle to Sybase ASE migration and needs special attention.

5.8.2.7.2 READ CONSISTENCY

Read consistency is achieved by following the ANSI standard for transaction isolation called SERIALIZATION. Oracle supports this natively and also adds its own interpretation of this standard and calls it READ ONLY. Sybase ASE implements read consistency with adding the HOLDLOCK parameter to each query and this can be achieve automatically by setting the isolation level 3.

Oracle

Sybase ASE

Oracle	Sybase ASE
<p>Read consistency as supported by Oracle does the following:</p> <ul style="list-style-type: none"> Ensures that the set of data seen by a statement is consistent at a single point-in-time and does not change during statement execution Ensures that reads of database data do not wait for other reads or writes of the same data Ensures that writes of database data do not wait for reads of the same data Ensures that writes wait for other writes only if they attempt to update identical rows in concurrent transactions <p>To provide read consistency, Oracle creates a read-consistent set of data when a table is being read and simultaneously updated.</p> <p>Read consistency functions as follows:</p> <ol style="list-style-type: none"> When an update occurs, the original data values changed by the update are recorded in rollback segments. While the update remains part of an uncommitted transaction, any user that reads the modified data views the original data values. Only statements that start after another user's transaction is committed reflect the changes made by the transaction. 	<p>Sybase ASE provides the HOLDLOCK function for transaction-level read consistency or the ability to set isolation level 3 at the session level. For example, specifying a SELECT with HOLDLOCK puts a shared lock on the data. More than one user can execute a SELECT with HOLDLOCK at the same time without blocking each other.</p> <p>When one of the users tries to update the selected data, HOLDLOCK blocks the update until the other users commit, rollback, or attempt an update and a deadlock occurs. This means that HOLDLOCK prevents other transactions from updating the same data until the current transaction is in effect.</p>
<p>You can specify that a transaction be read only using the following command:</p> <pre>SET TRANSACTION READ ONLY</pre>	<p>Replace this with</p> <pre>Set isolation level 3</pre>

5.8.2.8 LOGICAL TRANSACTION HANDLING

Oracle transactions are always implicit, which means that individual statements are not committed automatically. Sybase ASE transactions are non-implicit, which means that transactions are committed automatically. In Sybase ASE by default, each insert, update, and delete statement is considered a single transaction.

Oracle

Statements are not automatically committed to the database. The COMMIT WORK statement is required to commit the pending changes to the database.

Oracle transactions are implicit. This means that the logical transaction starts as soon as data changes in the database.

COMMIT WORK commits the pending changes to the database.

ROLLBACK undoes all the transactions after the last COMMIT WORK statement. Savepoints can be set in transactions with the following command:

```
SET SAVEPOINT savepoint_name
```

The following command rolls back to the specified SAVEPOINT:

```
ROLLBACK <savepoint_name>
```

Two-phase commit is automatic and transparent in Oracle. Two-phase commit operations are needed only for transactions which modify data on two or more databases.

Sybase ASE

By default, statements are automatically committed to the database. As a result any statement not within a transaction is automatically committed. A statement can be a batch containing multiple T-SQL statements that are sent to the server as one stream. The changes to the database are automatically committed after the batch executes. A ROLLBACK TRAN statement subsequently executed has no effect. In Sybase ASE, transactions are not implicit.

Sybase ASE allows you to nest BEGIN TRAN/COMMIT TRAN statements. When nested, the outermost pair of transactions creates and commits the transaction. The inner pairs keep track of the nesting levels. A ROLLBACK command in the nested transactions rolls back to the outermost BEGIN TRAN level, if it does not include the name of the SAVEPOINT.

Use the following commands to create transactions:

- **begin transaction** – marks the beginning of the transaction block. The syntax is:

```
begin {transaction | tran}  
[transaction_name]
```

transaction_name is the name assigned to the transaction. It must conform to the rules for identifiers. Use transaction names only on the outermost pair of nested **begin/commit** or **begin/rollback** statements.

- **save transaction** – marks a savepoint within a transaction:

```
save {transaction | tran}  
savepoint_name
```

savepoint_name is the name assigned to the

Oracle	Sybase ASE
	<p>savepoint. It must conform to the rules for identifiers.</p> <ul style="list-style-type: none"> • commit – commits the entire transaction: <pre>commit [transaction tran work] [transaction_name]</pre> <ul style="list-style-type: none"> • rollback – rolls a transaction back to a savepoint or to the beginning of a transaction: <pre>rollback [transaction tran work] [transaction_name savepoint_name]</pre>
<p>When a CHECKPOINT occurs, the completed transactions are written to the database device. A CHECKPOINT writes all dirty pages to the disk devices that have been modified since last checkpoint</p>	<p>Completed transactions are written to the database device at CHECKPOINT. A CHECKPOINT writes all dirty pages to the disk devices since the last CHECKPOINT.</p> <p>Calls to remote procedures are executed independently of any transaction in which they are included.</p>

5.8.3 DATABASE SECURITY

The Security functions supported by Sybase ASE and Oracle:

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

Security audit – Both Oracle and Sybase ASE includes an audit system. This is a comprehensive audit mechanism that checks access, authentication attempts, and administrator functions. The security audit records the date, time, responsible individual and other details describing the event in the audit trail.

An audit trail can be used to detect penetration of the system and misuse of resources. By examining the audit trail, a System Security Officer can inspect patterns of access to objects in databases and can monitor the activity of specific users. Audit records are traceable to specific users, which may act as a deterrent to users who are misusing the system.

In Sybase ASE, System Security Officer is the only user, who can start and stop auditing, set up auditing options, and process the audit data. A System Security Officer, you can establish auditing for events such as:

- Server-wide, security-relevant events
- Creating, deleting, and modifying database objects
- All actions by a particular user or all actions by users with a particular role active
- Granting or revoking database access
- Importing or exporting data
- Logins and logouts

User data protection – Both Oracle and Sybase ASE implements the discretionary access control policy over applicable database objects: databases, tables, views, and stored procedures.

It provides access controls that give object owners the ability to restrict access to objects, usually with the grant and revoke commands. This type of control is dependent upon an object owner's discretion.

Division of Roles – Both Oracle and Sybase ASE allows an administrator to grant privileged roles to the specified users so only designated users can perform certain tasks. Both have the predefined system roles. Sybase ASE has predefined roles, called "system roles," such as System Administrator and System Security Officer. In addition, Sybase ASE allows System Security Officers to define additional roles, called "user-defined roles."

Protection of the TSF – Sybase ASE 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 Sybase ASE have the appropriate access settings. Sybase ASE interacts with users through well-defined interfaces designed to ensure that its security policies are enforced.

Confidentiality of data – Both Oracle and Sybase ASE provides the data encryption for network protection. Sybase ASE allows maintaining the confidentiality of data by encrypting client-server communications using the secure socket layer (SSL) standard or using Kerberos.

Resource utilization – Sybase ASE provides resource limits to prevent queries and transactions from monopolizing server resources. This is similar to Oracle Resource Manager.

Password protected database dump and load – Protects database dump from unauthorized loads using the **password** parameter of the **dump database** command. If you include the **password** parameter when you make a database dump, you must also include this password when you load the database.

5.8.4 TRANSACTION PROCESSING IN STORED PROCEDURES

Sybase ASE 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 ASE 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, Sybase ASE 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. Sybase ASE determines what state to return by keeping track of any transaction changes after a statement executes. Sybase ASE 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_procxmode procedure_name, "chained"
```



This will allow transactions to stay open across multiple stored procedures. This is the ANSI-standard behavior and most likely required with any Oracle application using stored procedures.

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

```
sp_procxmode 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.

6 DBA OPERATIONS GUIDE

6.1 TRANSACTIONS

Transactions are one of the features that set a database apart from a file system. The main purpose of transactions in the database is to take the database from one consistent state to the next.

Transactions in both, Oracle and Sybase ASE, exhibit all of the required ACID characteristics. ACID is an acronym for:

- **Atomicity:** Either all of the transaction happens or none of it happens.
- **Consistency:** A transaction takes the database from one consistency state to the next.
- **Isolation:** The effects of a transaction may not be visible to other transactions until the transaction is committed.
- **Durability:** Once the transaction is committed, it is permanent.



Because Oracle's transactions are implicit and Sybase ASE's are non-implicit, there is a fundamental difference that needs to be addressed during the migration process. You might think that the solution would be to add a BEGIN TRANSACTION statement at the beginning of every process. But this is only half of the equation, Oracle's non-blocking reads allows for long running transactions. This is one area in Sybase ASE you need to pay close attention to, long running transactions and their impact on blocking read access.

One important part of an Oracle to Sybase ASE migration is to analyze the transactions and develop a strategy and implementation plan to re-package these transactions. Implementing datarow-locked tables when migrating the schema is one approach, but should be accompanied by a transaction analysis.



One important function a DBA needs to perform after an Oracle to Sybase ASE migration is to monitor blocked sessions very closely. The system tables `sysprocesses` and `syslogshold` will tell you who is being blocked and for how long the locks have been held by the blocking session.

6.2 REDO AND UNDO

In an Oracle database, redo and undo are two of the most important pieces. *Redo* is the information Oracle records in online, and archived, redo log files in order to “replay” your transaction in the event of a failure. Undo is the information Oracle records in the undo segments in order to reverse, or roll-back, your transaction.

6.2.1 REDO LOG

There are 2 types of redo log files:

- **Online:** These are redo log files actively maintained by the Oracle server. There are at least 2 online redo log groups with at least a single redo log file in each group. These files are maintained in a circular fashion, meaning the oldest redo log file will be overwritten by the latest log information.
- **Archived:** The ARCH process makes a copy of the oldest redo log file before it gets overwritten by the latest log information. This is a necessary step to preserve redo log information between full backups. Without archiving this information will be lost and no full recovery is possible.

With the introduction of the flashback technology in Oracle, the need of using online and archived redo log files to recover accidental drops of entire tables has been decreased. However, these files are crucial for a complete database recovery.



Sybase ASE maintains its redo information in the transaction log. Every Sybase ASE database has its own transaction log. The transaction log contains **committed** and **uncommitted** transactions and a truncation pointer that marks the beginning of the longest and open transaction in the transaction log. Unlike Oracle, Sybase ASE’s transaction log is not circular. This means when the transaction log gets full, intervention is necessary or that a threshold is setup to automatically dump the transactions and truncate the log.

There are several interventions available. Both of which can be setup to occur automatically via database thresholds:

- **Backup Transaction Log:** With the command `dump transaction [database name]` Sybase ASE issues a backup of all completed transactions in the transaction log prior the truncation

point, and then frees this space for future transactions. This is similar to the `ARCH` process in Oracle to create archived redo log files.

- **Truncate Transaction Log:** Instead of preserving transactions prior the truncation point, you can simply free the space for future transactions with the `dump transaction [database name] with truncate_only` command. This is an option that can be set at the database level to execute this behavior on an interval.



Unlike Oracle, Sybase ASE does not have a flashback technology, therefore access to a complete set of transaction log backups and the last full database backup are essential. Sybase ASE calls backup files database dumps and transaction dumps. This stems from Sybase ASE's backup command: `dump`.

As a rule of thumb, the transaction log size should be about 10-15% of the database size. In very large database settings a smaller transaction log size is sufficient. It all depends on the volume of data that gets changed. To enable up to the last transaction recovery the transaction log device can be mirrored. This is an effective way to protect against hardware failures and gain maximum recoverability.

Yes, Oracle DBAs have to adopt the concept of "transaction log full" situations. This is not as taunting as it may sound and is a non-issue in many cases. Added benefits are that every Sybase ASE database backup is a hot backup and no additional settings, configurations or setups have to be involved. This makes Sybase ASE backup and restores as easy and straight forward as it gets.

6.2.2 UNDO

Undo information is generated by the Oracle database as you make modifications to data to put it back the way it was before the modifications, in the event the transaction or statement you are executing fails for any reason or you request it with a `ROLLBACK` statement. "Rollback segment" and "undo segment" are considered synonymous terms.

Oracle stores the undo information in the undo tablespace or undo segments and protects itself from system failures by logging this information into the redo log files as well.



Sybase ASE maintains the undo information in the transaction log. Because it is the same transaction log that protects against system failures and provides recoverability up to the last transaction, no additional measures are needed to protect the undo information.

If for any reason a Sybase ASE transaction fails or you issue a `rollback transaction` command, the data will be put back as it was before you made the modifications.

6.3 TABLES AND INDEXES

6.3.1 TABLES

Oracle supports 9 major table types that support the applications in many different ways. The following table illustrates which Sybase ASE table and index combination most closely aligns to the Oracle table types, although there is no exact equivalent available in Sybase ASE for any of the 9 major Oracle table types.

Oracle	Sybase ASE
Heap organized tables: This is the Oracle default, row locking table construct.	A table created with the <code>lock datarows</code> clause.
Index organized tables: These tables are stored in an index structure, according to the primary key.	Creating a table with the <code>lock allpages</code> clause and creating a clustered index on the primary key.
Index clustered tables: These are clusters of one or more tables stored on the same database block. The data is “clustered” around the cluster key value.	Creating a table with the <code>lock allpages</code> clause and using the <code>partition</code> clause to partition the table across segments. Plus creating a clustered index on the primary key.
Hash clustered tables: Similar to clustered tables but instead of using a B*Tree index for the key the hash cluster hashes the key to the cluster. The data is the index.	Creating a table with the <code>lock allpages</code> clause and creating a clustered index on the primary key.
Sort hash clustered tables: Similar to the hash clustered table, but used on timestamp-based records using the first in first out (FIFO) approach.	Creating a table with the <code>lock allpages</code> clause and creating a clustered index on a timestamp. No FIFO support.

Oracle	Sybase ASE
Nested tables: These are system-generated and – maintained child tables in a parent/child relationship. These are not “stand-alone” tables.	N/A
Temporary tables: These are tables that store data for the life a transaction or the life of a session. Data is only visible to the session that created the temporary table:	Temporary tables in the tempdb that are identified by starting with a # sign. These tables are created by the session with either create table #temp_table or select ... into #temp_table command.
Object tables: These tables contain special attributes not available with non-object tables such as system generated REF (object identifier) for each row. This is a special case of heap, index organized and temporary tables.	N/A
External tables: These tables are not stored in the database itself. These are essentially files residing outside the database, but are made available to the database query engine as if they are tables. Very powerful in loading data from external sources.	Sybase ASE provides proxy tables with the Component Integration Services (CIS). This allows access to external data in files on filesystems or tables in other, non-Sybase ASE database systems.

6.3.2 INDEXES

Oracle maintains 5 major index types to support its data store and to improve query performance during data retrieval. Sybase ASE supports 3 major index types.

Oracle	Sybase ASE
B*Tree Index: This is the most commonly used index in Oracle. It is similar to a binary tree index, but Oracle refers to it as a balanced tree index. There several sub types of this index type like index organized tables, B*Tree clustered tables, descending indexes and reverse key indexes.	Non-Clustered Index: This is the most commonly used index in Sybase ASE. This is a B*Tree index like Oracle. It includes root level, leaf level and intermediate level entries to fast access the indexed data.
N/A	Clustered Index: Clustered indexes on allpages locked tables maintain the index key data in key order. The index is the data. You can create only one clustered index per table.
Bitmap Index: With B*Tree indexes there is a one-to-one	N/A

Oracle	Sybase ASE
relationship between a data row and an index entry pointing to this data row. A bitmap index has a single index entry pointing to many data rows simultaneously. This is very useful with tables having low cardinality data values in indexed columns. For example a bitmap index on the state code column.	
Bitmap Join Index: This index allows joining other tables to an indexed column in the main table. This will eliminate the need to join tables at the query level.	N/A
Function-based Index: This is an index based on the result of a function. You can consider them an index on a virtual (or derived) column.	Function-based Index: This index can be used to provide an inexpensive option for enhancing the performance of certain legacy applications. Function-based indexes allow you to create indexes based directly on one or more Transact-SQL expressions.
Application domain index: These are indexes you build and store yourself, either in Oracle or perhaps even outside Oracle. The Oracle text index is an example of an application domain index. It uses a set of tables to implement its concept of an index.	N/A

6.4 SYSTEM VIEWS

Oracle maintains system information in so called system views. There are 2 different types of views, the static data dictionary views and the dynamic performance views or so called v\$ views.

6.4.1 ORACLE STATIC DATA DICTIONARY VIEWS

Data dictionary tables are not directly accessible, but you can access information in them through data dictionary views. To list the data dictionary views available to you, query the view **DICTIONARY**.

Many data dictionary tables have three corresponding views:

- An **ALL_** view displays all the information accessible to the current user, including information from the current user's schema as well as information from objects in other schemas, if the current user has access to those objects by way of grants of privileges or roles.
- A **DBA_** view displays all relevant information in the entire database. DBA_ views are intended only for administrators. They can be accessed only by users with the SELECT ANY TABLE privilege. This privilege is assigned to the DBA role when the system is initially installed.
- A **USER_** view displays all the information from the schema of the current user. No special privileges are required to query these views.

The columns of the **ALL_**, **DBA_**, and **USER_** views corresponding to a single data dictionary table are usually nearly identical. Therefore, these views are described in full only once in this chapter, at their first occurrence alphabetically, and are listed without full descriptions at their other occurrences.

6.4.2 ORACLE DYNAMIC PERFORMANCE VIEWS

The actual dynamic performance views are identified by the prefix V_\$. Public synonyms for these views have the prefix V\$. Database administrators and other users should access only the V\$ objects, not the V_\$ objects.

The dynamic performance views are used by Oracle Enterprise Manager, which is the primary interface for accessing information about system performance. After an instance is started, the V\$ views that read from memory are accessible. Views that read data from disk require that the database be mounted, and some require that the database be open.

6.4.3 SYBASE ASE STATIC DATA DICTIONARY TABLES



Sybase ASE uses a combination of system tables and stored procedures to provide static data dictionary information. Unlike Oracle, Sybase ASE allows direct access to its data dictionary tables. To allow very experienced DBAs access to modify data dictionary information directly, Sybase ASE offers an overwrite switch that turns the option dynamically on and when the action is complete, turn it off again. This is very helpful with emergency and recovery activities.

The data dictionary tables are located in the master database, the sybsystemdb database and every user database.

System data dictionary categories and locations are as follows:

- Master Database
 - Configuration
 - User, Roles and Permissions
 - Storage
 - Time Ranges & Limits
 - Messages
 - Processes & Locks, Virtual Catalogs
 - Language & Character Sets
 - Remote Servers
 - Miscellaneous
- Sybsystemdb Database
 - Miscellaneous
- All Databases
 - User, Roles and Permissions
 - Objects
 - Integrity Relationships
 - Storage
 - Messages
 - Miscellaneous

In addition to the standard system tables included in all databases, the dbcc management database, dbccdb, contains seven tables that define inputs to and outputs from dbcc checkstorage. It also contains at least two workspaces.

6.4.4 SYBASE ASE DYNAMIC PERFORMANCE TABLES



Sybase ASE provides access the low-level monitoring information through proxy tables called "MDA tables". These proxy tables are essentially pointing the system owned \$ views. MDA is short for "Monitoring Data Access".

The MDA tables cover the following categories:

- Cache information
- Logical and physical I/O usage
- Java information
- Process activities
- Query Plans and Query Plan Execution
- System Waits
- System information specific to cluster environments

6.4.5 STATIC DATA DICTIONARY COMPARISON

This is a very limited sample of DBA tasks that provide a side-by-side comparison of Oracle and Sybase ASE system information retrieval tasks..

Oracle	Sybase ASE
Retrieve information about all tables, views and tables columns. <pre>SELECT * FROM ALL_TABLES SELECT * FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='tablename'</pre>	Retrieve information about all database objects like tables, views, stored procedures and triggers. Execute stored procedure: <code>sp_help</code> Retrieve column and detail storage information on each object. Execute stored procedure: <code>sp_help <object name></code>
Retrieve information about current sessions and wait status, use the following command: <pre>SELECT NVL(a.username, '(oracle)') AS username, a.osuser, a.sid, a.serial#,</pre>	Retrieve information about the current process and session status: Execute stored procedure: <code>sp_who</code>

Oracle	Sybase ASE
<pre> d.spid AS process_id, a.wait_class, a.seconds_in_wait, a.state, a.blocking_session, a.blocking_session_status, a.module, TO_CHAR(a.logon_Time, 'DD-MON-YYYY HH24:MI:SS') AS logon_time FROM v\$session a, v\$process d WHERE a.paddr = d.addr AND a.status = 'ACTIVE' ORDER BY 1,2; </pre>	
<p>Retrieve information about all database sessions with the username displayed as a hierarchy if locks are present:</p> <pre> SELECT LPAD(' ', (level-1)*2, ' ') NVL(s.username, '(oracle)') AS username, s.osuser, s.sid, s.serial#, s.lockwait, s.status, s.module, s.machine, s.program, TO_CHAR(s.logon_Time, 'DD-MON-YYYY HH24:MI:SS') AS logon_time FROM v\$session s CONNECT BY PRIOR s.sid = s.blocking_session START WITH s.blocking_session IS NULL; </pre>	<p>Retrieve the object names and IDs of processes that currently hold locks.</p> <p>Execute the stored procedure: <code>sp_lock</code></p>
<p>Retrieve high level information about the entire server installation:</p> <pre> SELECT * FROM v\$database; </pre>	<p>Comparable system calls are:</p> <p>Execute stored procedure: <code>sp_helpdb</code></p>
<pre> SELECT * FROM v\$instance; </pre>	<p>Execute stored procedure: <code>sp_server_info</code></p>
<pre> SELECT * FROM v\$version; </pre>	<p><code>select @@version</code></p>
<pre> SELECT a.name, </pre>	<p>Execute stored procedure: <code>sp_configure 'mem'</code></p>

Oracle	Sybase ASE
<pre> a.value FROM v\$sga a; </pre>	
<pre> SELECT Substr(c.name,1,60) "Controlfile", NVL(c.status,'UNKNOWN') "Status" FROM v\$controlfile c ORDER BY 1; </pre>	N/A
<pre> SELECT Substr(d.name,1,60) "Datafile", NVL(d.status,'UNKNOWN') "Status", d.enabled "Enabled", LPad(To_Char(Round(d.bytes/1024000,2), '9999990.00'),10,' ') "Size (M)" FROM v\$datafile d ORDER BY 1; </pre>	Execute stored procedure: sp_helpdevice
<pre> SELECT l.group# "Group", Substr(l.member,1,60) "Logfile", NVL(l.status,'UNKNOWN') "Status" FROM v\$logfile l ORDER BY 1,2; </pre>	Execute stored procedure: sp_helpdb <databasename>

6.4.6 DYNAMIC PERFORMANCE VIEWS COMPARISON

These are a couple of simple performance view lookups to demonstrate the differences. Oracle is using v\$ tables for all its performance view lookups. Sybase ASE uses MDA tables and stored procedures in combinations.

Oracle	Sybase ASE
<p>Retrieve a query plan (explain plan) of a running session in a hierarchical tree display.</p> <pre> SELECT LPAD(' ', 2 * (level - 1)) DECODE (level,1,NULL,level-1 ' .' pt.position ' ') INITCAP(pt.operation) DECODE(pt.options,NULL,', ' (' INITCAP(pt.options) ' ')) plan, pt.object_name, pt.object_type, pt.bytes, pt.cost, pt.partition_start, pt.partition_stop FROM plan_table pt START WITH pt.id = 0 AND pt.statement_id = '<session ID>' CONNECT BY PRIOR pt.id = pt.parent_id AND pt.statement_id = '<session ID>'; </pre>	<p>Comparable system call is:</p> <p>Execute stored procedure: sp_showplan <spid></p>
<p>Retrieve the top 10 sessions that use the most resources:</p> <pre> SELECT * FROM (SELECT Substr(a.sql_text,1,50) sql_text, Trunc(a.disk_reads/Decode(a.executions,0,1 ,a.executions)) reads_per_execution, a.buffer_gets, a.disk_reads, a.executions, a.sorts, a.address FROM v\$sqlarea a ORDER BY 2 DESC) WHERE rownum <=10; </pre>	<p>Comparable system call against a MDA table is:</p> <pre> select top 10 * from master..monProcessObject order by PhysicalReads desc </pre>

Oracle	Sybase ASE
Retrieve the most current SQL statement from an active session:	Comparable system call against a MDA table is:
<pre>SELECT a.sql_text FROM v\$sqltext_with_newlines a WHERE a.address = UPPER('<a.address from query above') ORDER BY a.piece;</pre>	<pre>Select * from master..monProcessSQLText where SPID = <spid from query above></pre>

Oracle maintains hundreds of data dictionary and performance views that can be queried in endless combinations. This is a very flexible approach to provide as much information to DBAs as possible. Sybase ASE provides a little bit over a hundred stored procedures to support DBAs in retrieving the data from the data dictionary. If these stored procedures don't provide the desired results, DBAs have the option to query the data dictionary directly. Retrieving performance information from Sybase ASE's MDA tables is as flexible as querying Oracle's v\$ views.

6.5 PARALLEL EXECUTION

The introduction of parallel execution was a blessing for many DBAs who were battling smaller maintenance windows and larger data volumes. Larger data volumes also created the need to get the job done in the same or shorter time and parallel execution was the key. Before diving into the key differences between Oracle and Sybase ASE parallel execution, here's a word of caution. Both databases recommend to not using parallel execution on small tasks like OLTP systems with short transactions or if the available resources are already highly utilized. If either one of these conditions holds true, parallel execution could have a negative effect on your database performance. Both database systems allow selectively enabling or disabling parallel execution.

The table below illustrates how Oracle and Sybase ASE manage parallel execution. This is mainly for informational purposes and should provide enough content for you to understand the differences.

Oracle	Sybase ASE
Parallel Query: The ability to perform a single query using many operating system processes or threads. Oracle will find operations it can perform in parallel, such as full	Parallel Query: Sybase ASE supports vertical, horizontal and pipelined parallel query execution. Vertical parallelism is the ability to run multiple operators simultaneously by

Oracle

table scans or large sorts, and create a query plan to do so.

By default the Oracle Database is enabled for parallel execution and when Automatic Degree of Parallelism (Auto DOP) is active, the database automatically decides if a statement should execute in parallel or not and what DOP it should use.

- 1) Enable the table(s) for parallel execution:

```
alter table sales parallel;  
alter table customers parallel;
```

Use this method if you generally want to execute operations accessing these tables in parallel.

- 2) Use a parallel hint in the SQL statement:

```
SELECT /*+ parallel(c) parallel(s) */  
c.state_province, sum(s.amount) revenue  
FROM customers c, sales s  
WHERE s.cust_id = c.cust_id  
AND s.purchase_date BETWEEN  
TO_DATE('01-JAN-2010','DD-MON-YYYY')  
AND TO_DATE('31-DEC-2010','DD-MON-  
YYYY');
```

This method is mainly useful for testing purposes, or if you have a particular statement or few statements that you want to execute in parallel, but most statements run in serial.

- 3) Use an alter session command:

```
alter session force parallel query ;
```

Sybase ASE

employing different system resources such as CPUs, disks, and so on. Horizontal parallelism is the ability to run multiple instances of an operator on the specified portion of the data. The way you partition your data greatly affects the efficiency horizontal parallelism.

Pipelining is a form of vertical parallelism in which intermediate results are piped to higher operators in a query tree. The output of one operator is used as input for another operator. The operator used as input can run simultaneously with the operator feeding the data, which is an essential element in pipelined parallelism.

To enable parallelism on Sybase ASE the following configuration parameters have to be configured:

- **number of worker processes**
- **max parallel degree**
- **max resource granularity**
- **max repartition degree**
- **max scan parallel degree**
- **prod-consumer overlap factor**
- **min pages for parallel scan**
- **max query parallel degree**

After Sybase ASE is configured for parallel query processing, the query optimizer evaluates each query to determine whether it is eligible for parallel execution. If it is eligible, and if the optimizer determines that a parallel query plan can deliver results faster than a serial plan, the query is divided into plan fragments that are processed simultaneously.

The level of parallel query execution can be set server wide or session specific through the set command:

```
Set parallel_degree 5
```

Or by using a SQL statement extension to control parallel query execution at the SQL statement level:

```
select ...  
from tablename [( [index index_name]  
[parallel [degree_of_parallelism | 1 ]]  
[prefetch size] [lru|mrul] ) ] ,
```

Oracle

Sybase ASE

```
tablename [( [index index_name]
[parallel [degree_of_parallelism | 1]
[prefetch size] [lru|mru] ) ] ...
```

Parallel DML: Data Manipulation Language (DML) operations such as INSERT, UPDATE, and DELETE can be parallelized by Oracle. Parallel execution can speed up large DML operations and is particularly advantageous in data warehousing environments where it's necessary to maintain large summary or historical tables. In OLTP systems, parallel DML sometimes can be used to improve the performance of long-running batch jobs.

When you issue a DML statement such as an INSERT, UPDATE, or DELETE, Oracle applies a set of rules to determine whether that statement can be parallelized. For UPDATE and DELETE statements, the rules are identical. INSERT statements, however, have their own set of rules.

Rules for UPDATE and DELETE statements

- Oracle can parallelize UPDATE and DELETE statements on partitioned tables, but only when multiple partitions are involved.
- You cannot parallelize UPDATE or DELETE operations on a non-partitioned table or when such operations affect only a single partition.

Rules for INSERT statements

- Standard INSERT statements using a VALUES clause cannot be parallelized.
- Oracle can parallelize only INSERT ... SELECT ... FROM statements.

Parallel DDL: Parallel DDL works for both tables and indexes, whether partitioned or non-partitioned.

For non-partitioned tables and indexes, only the following types of DDL statements can be parallelized:

Parallel DML: Sybase ASE does not support parallel execution on INSERT, UPDATE and DELETE statements. However, tables other than the destination table used in the query's FROM clause can be accessed in parallel.

Parallel DDL: Sybase ASE supports parallel DDL on index creation only. To enable parallel index creation the table needs to be partitioned. When creating an index the data sorting part will be executed in parallel.

Oracle

```
CREATE TABLE...AS SELECT
CREATE INDEX
ALTER INDEX...REBUILD
```

If you're working with partitioned tables and indexes, the scope of Oracle's parallel DDL support broadens. The following statements can be parallelized for partitioned tables and indexes:

```
CREATE TABLE...AS SELECT
ALTER TABLE...MOVE PARTITION
ALTER TABLE...SPLIT PARTITION
CREATE INDEX
ALTER INDEX...REBUILD PARTITION
ALTER INDEX...SPLIT PARTITION
```

Parallel Data Loading: Oracle's SQL*Loader utility loads data into Oracle tables from external files. With some restrictions, SQL*Loader supports the loading of data in parallel. If you have a large amount of data to load, SQL*Loader's parallel support can dramatically reduce the elapsed time needed to perform that load.

SQL*Loader supports parallel loading by allowing you to initiate multiple concurrent direct path load sessions that all load data into the same table or into the same partition of a partitioned table. Unlike the case when you execute a SQL statement in parallel, the task of dividing up the work falls on your shoulders. Follow these steps to use parallel data loading:

- Create multiple input datafiles.
- Create a SQL*Loader control file for each input datafile.
- Initiate multiple SQL*Loader sessions, one for each control file and datafile pair.

Sybase ASE

Parallel execution is automatically enabled on partitioned tables and the following commands:

```
create index ...
update statistics
```

Parallel Data Loading: Sybase ASE loads data from external files with the bcp command. Loading data into a partitioned table using parallel bcp lets you direct the data to a particular partition in the table.

- Before you run parallel bulk copy, the table should be located on the segment, and it should be partitioned.
- You should drop all indexes, so that you do not experience failures due to index deadlocks.
- Use alter table...disable trigger so that fast, minimally-logged bulk copy is used, instead of slow bulk copy, which is completely logged.
- You may also want to set the database option trunc log on chkpt to keep the log from filling up during large loads.
- You can use operating system commands to split the file into separate files, and then copy each file, or use the -F (first row) and -L (last row) command-line flags for bcp.

Whichever method you choose, be sure that the number of rows sent to each partition is approximately the same.

Parallel Recovery: Parallel recovery can speed up both

Parallel Recovery: Sybase ASE supports parallel recovery

Oracle

instance recovery and media recovery. In parallel recovery, multiple parallel slave processes are used to perform recovery operations. The SMON background process reads the redo log files, and the parallel slave processes apply the changes to the datafiles.

In a serial recovery scenario, the SMON background process both reads the redo log files and applies the changes to the datafiles. This may take a considerably long time when multiple datafiles need to be recovered. However, when parallel recovery is being used, the SMON process is responsible only for reading the redo log files. The changes are applied to the datafiles by multiple parallel slave processes, thereby reducing the recovery time.

Recovery requires that the changes be applied to the datafiles in exactly the same order in which they occurred. This is achieved by single-threading the read phase of the recovery process by the SMON process. SMON reads the redo log files and serializes the changes before dispatching them to the parallel slave processes. The parallel slave processes then apply those changes to the datafiles in the proper order. Therefore, the reading of the redo log files is performed serially even during a parallel recovery operation.

Specifying the RECOVERY_PARALLELISM Parameter

The RECOVERY_PARALLELISM initialization parameter controls the degree of parallelism to use for a recovery. You can override that setting for a specific situation by using the RECOVER command's PARALLEL clause.

A value of 0 indicates serial recovery, no parallelism will be used. The RECOVERY_PARALLELISM parameter setting cannot exceed the PARALLEL_MAX_SERVERS setting.

Sybase ASE

during start-up and failover, allowing databases to be recovered in parallel by multiple recovery tasks. Database recovery is an I/O-intensive process. The time to recover Sybase ASE with parallel recovery depends on the bandwidth of the underlying I/O subsystem. The I/O subsystem should be able to handle Sybase ASE concurrent I/O requests.

With parallel recovery, multiple tasks recover user databases concurrently. The number of recovery tasks is dependent on the configuration parameter max concurrently recovered db. The default value of 0 indicates that the Sybase ASE server adopts a self-tuning approach in which it does not make any assumptions on the underlying storage architecture. Statistical I/O sampling methods determine the optimal number of recovery tasks depending on the capabilities of the underlying I/O subsystem. An advisory on the optimal number of recovery tasks is provided. If the configuration value is nonzero, Sybase ASE spawns as many tasks as indicated by the configuration parameter and also by the number of open databases parameter.

During parallel recovery, the system administrator can force serial recovery by setting max concurrently recovered db to 1. The active recovery tasks drain out after completing the recovery of the database that is being worked on. The remaining databases are recovered serially.



Parallel execution is widely supported by both Oracle and Sybase ASE. It is a matter of syntax and becoming aware of certain restrictions when migrating from Oracle to Sybase ASE.

6.6 DATA LOADING AND UNLOADING

One of the key elements of an Oracle to Sybase ASE migration is the data unloading and loading. For completeness the data loading mechanism in Oracle are outlined as well, although these features do not play a role when migrating from Oracle to Sybase ASE.

6.6.1 DATA LOADING

Oracle	Sybase ASE
<p>SQL*Loader: This is Oracle's high-speed loader. It has two modes of operations:</p> <ul style="list-style-type: none"> • <i>Conventional Path:</i> SQL*Loader will employ SQL inserts on our behalf to load data. • <i>Direct Path:</i> SQL*Loader does not use SQL in this mode; it formats database blocks directly. <p>SQL*Loader is a one way method of data transfer. It loads data into Oracle very fast, but there's no option of unloading.</p> <p>The direct path load allows you to read data from a flat file and write directly to formatted database blocks, bypassing the entire SQL engine, undo generation and, optionally, redo generation at the same time.</p> <p>Parallel direct path load is among the fastest way to load an Oracle database.</p>	<p>bcp in: bcp provides a convenient, high-speed method for transferring data between a database table or view and an operating system file.</p> <p>bcp in works in one of two modes:</p> <ul style="list-style-type: none"> • <i>Slow bcp</i> – logs each row insert that it makes, used for tables that have one or more indexes. • <i>Fast bcp</i> – logs only page allocations, copying data into tables without indexes or at the fastest speed possible. You can use fast bcp on tables with non-clustered indexes. <p>Fast bcp will bypass the transaction logs and therefore is much faster, but also non-recoverable in case something goes wrong with the bcp load.</p> <p>bcp is the main tool to load data from operating system files into Sybase ASE. It is also used to move data between databases and to unload data from Sybase ASE.</p> <p>bcp is a standalone program executed from the operating system. If you need more flexibility of how bcp works, there is an API through the Client-Library that supports many languages.</p> <p>bcp fully supports parallel loading, which makes Fast-bcp even faster.</p> <p>These are the steps to load data with the Fast-bcp option:</p> <ul style="list-style-type: none"> • Use sp_dboption to set select into/bulkcopy/pllsort to true. • Run checkpoint in the database that was

Oracle

Sybase ASE

External Tables: They allow us to treat operating system files as if it's a read-only database table. Unless you have to load data over a network, massive multi user access is required or you need to load LOB datatypes, this is the preferred method.

External tables are created using the SQL CREATE TABLE . . . ORGANIZATION EXTERNAL statement. When you create an external table, you specify the following attributes:

- TYPE - specifies the type of external table. The two available types are the ORACLE_LOADER type and the ORACLE_DATAPUMP type. Each type of external table is supported by its own access driver.
 - The ORACLE_LOADER access driver is the default. It can perform only data loads, and the data must come from text datafiles. Loads from external tables to internal tables are done by reading from the text-only datafiles in the external table.
 - The ORACLE_DATAPUMP access driver can perform both loads and unloads. The data must come from binary dump files. Loads to internal tables from external tables are done by

changed.

- Have enough space to re-create any indexes on the table.
- Drop the indexes on the table.
- Have insert permission on the table.
- Perform the copy with bcp.
- Re-create the indexes.
- Reset sp_dboption, if desired, and run checkpoint in the database that was changed.
- Use dump database to back up the newly inserted data.
- Run stored procedures or queries to determine whether any of the newly loaded data violates rules.

CIS Proxy Tables: Component Integration Services (CIS) presents tables to a client application as if all the data in the tables were stored locally. Remote tables are mapped to local proxy tables which hold metadata. Internally, when a query involving remote tables is executed, the storage location is determined, and the remote location is accessed so that data can be retrieved.

The access method used to retrieve remote data is determined by two attributes of the external object:

- The server class associated with the remote object
- The object type

To achieve location transparency, tables must first be mapped to their corresponding external locations.

A server class must be assigned to each server when it is added using sp_addserver. Server classes determine the access method used to interact with the remote server. The server classes are:

- *ASEnterprise* – used if the server is Sybase ASE. This is the default server class.
- *ASAnywhere* – used if the server is Adaptive Server Anywhere version 6.0 or later. This server class should be used for Sybase IQ versions earlier than Sybase IQ 12.5.
- *ASIQ* – used if the server is Sybase IQ version

Oracle

fetching from the binary dump files.

Unloads from internal tables to external tables are done by populating the binary dump files of the external table.

- **DEFAULT DIRECTORY** - specifies the default location of files that are read or written by external tables. The location is specified with a directory object, not a directory path.
- **ACCESS PARAMETERS** - describe the external data source and implements the type of external table that was specified. Each type of external table has its own access driver that provides access parameters unique to that type of external table.
- **LOCATION** - specifies the location of the external data. The location is specified as a list of directory objects and filenames. If the directory object is not specified, then the default directory object is used as the file location.

The following example shows the use of each of these attributes:

```
SQL> CREATE TABLE cust_load
2   (cust_number      CHAR(5),
3    cust_last_name   CHAR(20),
4    cust_first_name  CHAR(15))
5  ORGANIZATION EXTERNAL
6  (TYPE ORACLE_LOADER
7   DEFAULT DIRECTORY def_dir1
8   ACCESS PARAMETERS
9    (RECORDS DELIMITED BY NEWLINE
10    FIELDS (cust_number      CHAR(2),
11            cust_last_name   CHAR(18),
12            cust_first_name  CHAR(11))
13   )
14  LOCATION ('cust.dat')
15  );
```

Table created.

Sybase ASE

12.5 and later.

- *local* – the local server. There can be only one.
- *direct_connect* – indicates that the server is an Open Server™ application that conforms to the interface requirements of a DirectConnect™ server. For access to Microsoft SQL Server, DB2, Oracle, or Informix, you must use a DirectConnect server.
- *sds* – indicates that the server conforms to the interface requirements of a Specialty Data Store.
- *RPCServer* – indicates that a server can only handle RPCs. It does not accept SQL statements, transaction control statements, or anything else.

The server presents a number of object types to client applications as if they were local tables. Supported object types are:

- *table* – the object in a remote server of any class is a relational table. This is the default type.
- *view* – the object in a remote server of any class is a view. Component Integration Services treats views as if they were local tables without any indexes.
- *remote procedure* – the object in a remote server of any class is a remote procedure. Component Integration Services treats the result set from the remote procedure as a read-only table.
- *file* – the object is an individual file within a file system.
- *directory* – the object is a file system directory.

CIS proxy tables are a very flexible and fast way to load data from an external source into a Sybase ASE database.

Example:

To allow access to Oracle tables via CIS, the Enterprise Connect Data Access Option for Oracle must be installed. In order to create a proxy table for an Oracle table the Oracle server must be accessible as a remote server from within Sybase ASE.

The ECDA is a separate process, like the backup process, and needs to be started separately.

Oracle	Sybase ASE
	<p>Once the ECDA server is running, define the remote Oracle server:</p> <pre>sp_addserver ORACLEDC, direct_connect, ORACLEDC</pre> <p>In the Oracle database there is the following table:</p> <pre>example_table (id_num int, name varchar(30), phone varchar(20) null, birthdate date null)</pre> <p>In the Sybase ASE database the following proxy tables will be created:</p> <pre>create existing table example_table (id_num int, name varchar(30), phone varchar(20) null, birthdate smalldatetime null) external table at 'ORACLEDC.oradb..example_table'</pre> <p>Now you can access the example_table from isql as if the table is inside the Sybase ASE database.</p> <p>This is just a short version of what needs to be happening to allow remote database access via CIS and ECDA. However, once configured and tweaked, these are the basic steps.</p>

6.6.2 DATA UNLOADING

Oracle	Sybase ASE
<p>Flat File Unload: Although Oracle has great tools to load data from flat files, there is no feature available that unloads data from an Oracle database into flat files.</p> <p>There are tools like import/export and data pump in and out. But these tools export the data in binary form and are</p>	<p>bcp out: This is the data unload method of bcp. bcp out describes the direct of the data flow. In this case it is from Sybase ASE to a flat file.</p> <p>bcp out also lets you select source data through a view. This allows you to create denormalized data views from</p>

not readable by non-Oracle tools.

The last ditch resort for you if you want to unload data is to create your own PL/SQL utility that essentially spools the data from the database into a flat file. There are also 3rd party products available that provide the same functionality.

normalized source data as input for a data warehouse engine like Sybase IQ.

If Sybase ASE is enabled for row-level access, and you bulk-copy-out data, bcp copies out only the rows of data to which you have access. To copy out the entire table, you must first drop the access rules, then bcp out. Reinststate the access rules after you are done, if applicable.

With the row delimiter and record delimiter parameter, bcp can create output files that suite any needs.

bcp parameters can be supplied as in-line parameters in a command line or as a format file that contains more granular formatting information.

Data Pump Unload: This is the ORACLE_DATAPUMP attribute of external tables. Although Oracle supports the ORACLE_LOADER attribute for external tables, data can only be unloaded using the ORACLE_DATAPUMP attribute.

isql output-file: Creating an output file from a SQL statement is similar to a spool command in Oracle. However, you don't have to create a PL/SQL procedure to spool data; it is a simple parameter to the isql command.



This eliminates this option for any Oracle to Sybase ASE migration activities. The data pump export is in binary form and can only be read by the data pump utility.

```
isql -o <output filename>
```

6.7 MIGRATE THE DATABASE USERS

This section explains the migration of users from Oracle to Sybase ASE. The privileges can be granted to the users directly or through roles.

A schema owner in Oracle is a user with privileges to create objects. This is true in Sybase ASE as well. A user will have to be created in Sybase ASE 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 users of Oracle and Sybase ASE can be classified as administrative users, application users and schema/database owners.

- Administrative users are users with special roles, such as database administrator and Security administrator
- Application users are who manipulate data in the owning user's tables
- Database owner are users who create and maintain objects related to an application.

Sybase ASE has two levels of logins. In addition of instance login, Sybase ASE requires separate logins to be created for each database that the user needs to connect to.

To avoid confusion while referring these various logins:

- **Oracle** – Instance level – user or username
- **Sybase ASE** – Instance level – login
- **Sybase ASE** – database level – user

Login provides access to the instance of Sybase ASE, whereas the user controls privileges to objects inside the database.

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

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

6.7.1 CREATE USER ACCOUNT

Get the list of users with privileges on all the objects of the database by querying the Oracle database on tables ALL_TAB_PRIVS and ALL_COL_PRIVS.

System stored procedure sp_addlogin is used to create login name in the Sybase ASE. 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.

6.7.2 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 Sybase ASE enable you to enforce individual accountability. Sybase ASE 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. For more details of system roles and related tasks see Chapter 14 of System Administrator guide.

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 through 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]
```

The attributes can be decoded as follows:

- **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 Sybase ASE 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 Sybase ASE 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'.

6.8 BACKUP AND RESTORE

Every system should have a recovery plan in case of a disaster. Described here are some basic Sybase ASE backup and restore recommendations.

The following four actions will ensure that the data can always be restored:

Build dbcc checks into the regular backup/maintenance schedule to ensure that there are consistent, accurate backups available at all times.

Create a master database dump, the database master should be dumped right after installation and again when the user databases have been created and initialized or when new users are added to the system. The master database contains details about the structure of your entire database system. Its keeps track of the Sybase ASE databases, devices, and device fragments that make up those databases. Because Sybase ASE needs this information during recovery, it is crucial that you maintain an up-to-date backup copy of the master database at all times.

To ensure that your backup of master is always up to date, back up the database after each command that affects disks, storage, databases, or segments. This means you should back up master after performing any of the following procedures:

- Creating or deleting databases
- Initializing new database devices
- Adding new dump devices
- Using any device mirroring command
- Creating or dropping system stored procedures, if they are stored in master
- Creating, dropping, or modifying a segment
- Adding new Sybase ASE logins

To back up *master* to a tape device, start **isql** and enter the command, where *tape_device* is the name of the tape device (for example, */dev/rmt0*):

```
dump database master to "tape_device"
```

Save system table data and other configuration information, this is standard DBA support documentation. In a disaster recovery situation and while troubleshooting performance problems, it provides information on resources in use. This is the information that needs to be saved:

- Hardware Configuration
- Disk Layout and usage
- CPUs
- RAM
- Software Configuration
- Operating system version/release
- Patches
- ASE Configuration
- ASE Name
- IP Address
- Port Number
- Master Device
- Sybssystemprocs Device
- Language
- Character Set
- Sort Order
- Error Log File
- Backup Server Name
- Error Log File
- Server Map
- sp_configure
- sp_helpsegment (object placement)
- sp_helpdb
- sp_helpdevice
- select * from sysusages
- select * from sys.servers
- select * from master..sysobjects
- select * from sysremotelogins
- select * from syslogins
- select * from sysloginroles

- databases and owners information

Sybase ASE uses transactions to keep track of all database changes. Each database has its own transaction log, the system table syslogs. The transaction log automatically records every transaction issued by each user of the database.

The transaction log is a write-ahead log. When a user issues a statement that modifies the database, Sybase ASE writes the changes to the log. After all changes for a statement have been recorded in the log, they are written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page. At that time, it is written to disk.

The transaction log contains enough information about each transaction to ensure that it can be recovered. Create dump of user databases and log. These database and log dumps will be used to restore the database to a consistent state when it is determined that the data is invalid or after a disaster. Consider using "incremental backups" where a database dump could be done every week followed by a transaction log dump every day. If the database size grows quickly or the data needs to be able to be restored to a more recent state, these should be done more frequently (see System Administration guide, Chapter 3: System Administration for Beginners).

A full Sybase ASE backup makes a full copy of the database. If there is a full transaction log it may be considered to first truncate the log (dump transaction <dbname> with truncate_only). Successive transaction log dumps can be applied onto a full database backup.

Backup

```
dump database testdb to "<filename1>"
go
<system activity>
dump transaction testdb to "<filename2>"
go
<system activity>
dump transaction testdb to "<filename3>"
go
```

Restore

```
load database testdb from "<filename1>"
```

```
go
load transaction testdb from "<filename2>"
go
load transaction testdb from "<filename3>"
go
online database testdb
go
```

6.8.1.1 AUTOMATIC RECOVERY AFTER A SYSTEM FAILURE OR SHUTDOWN

Each time you restart Sybase ASE—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 Sybase ASE, it performs database recovery in this order:

1. Recovers *master*.
2. Recovers *sybssystemprocs*.
3. Recovers *model*.
4. Creates *tempdb* (by copying *model*).
5. Recovers *sybssystemdb*.
6. 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 Sybase ASE as soon as the system databases have been recovered, but they cannot access other databases until they have been recovered.

6.9 OTHER ADMINISTRATOR TASKS

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

- **isql**, a simple tool to access the server
- **sp_configure**, the system stored procedure to change the server configuration
- **sp_dboption**, the system stored procedure to change the database options
- **dbcc**, the database consistency checker commands to check the data consistency
- **'set' commands**, to set a wide variety of server, database, table and session options like:
 - chained (use of begin transaction to start an implicit transaction)
 - transaction isolation level (to choose for example to do dirty reads)
 - showplan, statistics io, statistics time (to monitor query performance)

Other DBA tasks, that needs to be performed routinely:

6.9.1 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.

6.9.2 THRESHOLD CHECKING

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



Remember, the Sybase ASE transaction log is not circular and will fill up over time. You need to implement maintenance tasks to manage the transaction logs. This can be done either via the `sp_dboption 'trunc tran on checkpoint'` command, which basically cleans out the transaction log every time a checkpoint is issued. Or there are threshold parameters via the `sp_addthreshold` stored procedure. This allows setting an action on what to do when the threshold is reached and also eases the load on the system by issuing this command only when needed.

6.9.3 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
 - REORG REBUILD – Faster than dropping and recreating the clustered index
 - 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)

6.9.4 REFRESHING THE TABLE STATISTICS

For the Sybase ASE optimizers 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 index 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.

6.9.5 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 Sybase ASE Monitor, used for monitoring server performance and other activities, Sybase Control Center (SCC) which is used to monitor server performance and other activities as well as 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.

6.10 ORACLE DBA TOOLS AND SYBASE DBA TOOLS COMPARED

As a DBA you get used to certain tools and where to find them. This includes environment variables as well as supporting and installation tools. The following table will illustrate which Sybase command is comparable a Oracle command and where you can find it.

Description	Oracle	Sybase ASE
Home Directory	\$ORACLE_HOME	\$SYBASE
Default Database/Instance	\$ORACLE_SID	\$DSQUERY
Command-line tool for SQL	SQL*Plus in \$ORACLE_HOME/bin/sqlplus	isql in \$SYBASE/OCS- 15_0/bin/isql

Description	Oracle	Sybase ASE
Import/export data	<p>imp / exp command or impdp /expdp command for data pump located in \$ORACLE_HOME/bin</p> <p>Oracle imports and exports the data and the DDL definitions, plus all other objects like type definitions, indexes, procedure and views.</p>	<p>For the data import and export: bcp command located in \$SYBASE/OCS-15_0/bin</p> <p>For the definition import and export: defncopy command in \$SYBASE/OCS-15_0/bin</p> <p>The indexes must be treated separately. To reverse engineering the DDL to be recreated in another environment: ddlgen command in \$SYBASE/ASEP/bin</p>
Create a new database	dbca command in \$ORACLE_HOME/bin	Sybase Central or SQL command create database
Create new network connection	netca command in \$ORACLE_HOME/bin	dsedit in \$SYBASE/OCS-15_0/bin
Setup new Oracle Enterprise Server (OEM) server	emca command in \$ORACLE_HOME/bin	<p>Sybase Central in combination with Sybase Control Center is equivalent to OEM.</p> <p>Setup Sybase Central: installed automatically when Sybase ASE is installed. Installs as part of the client installation.</p> <p>Setup Sybase Control Center: Install the software with the supplied Sybase Installer.</p>
Load data into the database	SQL*LOADER in \$ORACLE_HOME/bin/sqlldr	bcp command located in \$SYBASE/OCS-15_0/bin using bcp in

Description	Oracle	Sybase ASE
Start database server	<p>Manual:</p> <p>Open SQL*Plus as sysdba</p> <p>SQL> STARTUP</p> <p>Starts the instance, , mounts the database and opens the database.</p> <p>Script:</p> <p>dbstart command in \$ORACLE_HOME/bin</p> <p>Both commands are using the spfiles located in \$ORACLE_HOME/dbs in the following order:</p> <ol style="list-style-type: none"> 1. spfile\$ORACLE_SID.ora 2. spfile.ora 3. init\$ORACLE_SID.ora 	<p>startserver command in \$SYBASE/ASE-15_0/install</p> <p>startserver -f RUN_\$DSQUERY</p> <p>The RUN_\$DSQUERY file is the spfile equivalent.</p>
Start backup server	N/A	<p>startserver command in \$SYBASE/ASE-15_0/install</p> <p>startserver -f RUN_\$DSQUERY_BS</p> <p>The RUN_\$DSQUERY_BS file contains the startup parameters.</p>
Start monitor server	<p>emctl command in \$ORACLE_HOME/bin</p> <p>emctl start dbconsole</p>	<p>startserver command in \$SYBASE/ASE-15_0/install</p> <p>startserver -f RUN_\$DSQUERY_MS</p> <p>The RUN_\$DSQUERY_MS file contains the startup parameters.</p>

Description	Oracle	Sybase ASE
Show running processes	Unix: <pre>ps -aef grep \$ORACLE_SID</pre> Windows: <pre>pslist -d oracle</pre>	<pre>showserver</pre> command in <pre>\$SYBASE/ASE-15_0/install</pre>
Stop database server	Login to SQL*Plus and execute: <pre>SQL>shutdown</pre> For normal shut down <pre>SQL>shutdown immediate;</pre> For immediate shut down <pre>SQL>shutdown abort;</pre> For emergency shut down	Login via isql and execute the command: <pre>1>shutdown</pre> Without parameters the server will wait for all transactions to finish. Adding 'with nowait' will terminate all sessions and shut down the server immediately.
Stop backup server	N/A	Login via isql into the Sybase ASE database server and execute the command: <pre>1>shutdown</pre> <pre><Backup_Server_name></pre> Without parameters the server will wait for all transactions to finish. Adding 'with nowait' will terminate all sessions and shut down the server immediately. This must be happen before shutting down the Sybase ASE database server.

Description	Oracle	Sybase ASE
Start Job Scheduler	N/A	<p>Login via isql into the Sybase ASE database server and execute the command:</p> <pre> use sybmgmtdb go sp_sjobcontrol @name=NULL, @option="start_js" go </pre>
Stop Job Schedule	N/A	<p>Login via isql into the Sybase ASE database server and execute the command:</p> <pre> use sybmgmtdb go sp_sjobcontrol @name=NULL, @option="stop_js_wait" go </pre>

7 MIGRATING THE DATABASE ARCHITECTURE

The migration of the Oracle database architecture to Sybase ASE 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 Oracle database and transform into Sybase ASE database schema, run the Sybase ASE schema
2. Create the administration and security
 - describe the Oracle security model
 - translate this into Sybase ASE syntax
3. Migration of data
 - export the Oracle 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 Sybase ASE, 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 Oracle database.

7.1 BUILD THE SYBASE INSTANCE

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

7.1.1 PRE INSTALLING PLANNING

Following guidelines will help to install and configure Sybase ASE:

- 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 Sybase ASE 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 Sybase ASE to run, the operating system must be configured to allow allocation of a shared memory segment at least as large as the Sybase ASE **total logical memory** configuration parameter.

After you install Sybase ASE, 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 Sybase ASE, that device cannot be used for any other purpose (even if Sybase ASE never fills the device with data). Likewise, Sybase ASE 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.
- Sybase ASE 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.

7.1.2 INSTALLATION

The following installation options have to be determined before installing the Sybase ASE:

- Install directory: A directory for the installation of Sybase ASE.
- 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 Sybase ASE:
Choose the editions of Sybase ASE from:
 - Enterprise Edition
 - Small Business Edition
 - Developers Edition
- Choose servers to configured: A Full or Custom installation allows you to choose to
 - Configure new Sybase ASE
 - 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 Sybase ASE instance, following servers will be installed:

- Sybase ASE – <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

7.2 CONFIGURE THE SERVER

Installation of the Sybase ASE 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. When you change a configuration parameter, Sybase ASE saves a copy of the old configuration file as *server_name.001*, *server_name.002*, and so on. Sybase ASE 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.

In Oracle, the characteristics and behavior of the database and the instance are determined by a large set of parameters stored in the initialization file (init.ora) or server parameter file (spfile). These parameters cover a diverse set of resources, such as memory, processes, network, disk, I/O, connections, files, character set, and so on.

The non-default values of the Oracle initialization parameters can be obtained from the parameter file if one is in use. If a server parameter file is in use, the parameter values can be obtained using one of the following options:

- Convert the server parameter file (spfile) to an initialization parameter file by executing the following statement:
`CREATE pfile FROM spfile`
- Query the database by executing the following statement:
`SELECT name, value
FROM sys.v$spparameter
WHERE isspecified = 'TRUE'`

Configuration parameters are used for a wide range of services, from basic to specific server operations, and for performance tuning.

7.2.1 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 Sybase ASE behavior they affect. This makes it easier to identify all parameters that you might need to tune to improve a particular area of Sybase ASE performance.

Parameter group	Configures Sybase ASE 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

Parameter group	Configures Sybase ASE for:
Diagnostics	Diagnostic principles
Disk I/O	Disk I/O
Error log	Error log and the logging of Sybase ASE events to the Windows Event Log
Extended stored procedures	Affecting the behavior of extended stored procedures (ESPs).
General information	Basic system administration
Java services	<p>Memory for Java in Sybase ASE</p> <p>See the <i>Java in Adaptive Server Enterprise</i> manual for complete information about Java in the database.</p> <p>If you use method calls to JDBC, you may need to increase the size of the execution stack available to the user. See “stack size” for information about setting the stack size parameter.</p>
Languages	Languages, sort orders, and character sets
Lock manager	Locks
Memory use	Memory consumption
Meta-data caches	<p>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. For a discussion of metadata caches in a memory-usage context, see System Administrator Guide, Chapter 3: “Configuring Memory”.</p>
Monitoring	<p>Collecting monitoring information. By default, Sybase ASE does not collect monitoring information required by the monitoring tables.</p> <p>For more information about the monitoring tables see Chapter 2, “Monitoring Tables,” in the <i>Performance and Tuning Guide: Monitoring and Analyzing</i>.</p>
Network communication	Communication between Sybase ASE and remote servers, and between Sybase ASE and client programs
O/S resources	Use of operating system resources

Parameter group	Configures Sybase ASE 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 Sybase ASE 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.

7.2.2 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.

For more detail on changing the database options, see System Administrator guide, Chapter 8: "Setting Database Options".

7.3 MIGRATE THE STORAGE ARCHITECTURE

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



Most Oracle installations enlist the help of Oracle's Automated Storage Manager (ASM). An ASM equivalent is not available in Sybase ASE. Storage must be managed through the command line tool *isql* or via Sybase Central, the database admin GUI tool.

To an Oracle DBA, creating a database means creating an entire database system that contains control files; redo logs, data dictionary and temporary tablespace. In Sybase ASE, these tasks are accomplished as part of the installation process. Hence, creating a database in Sybase ASE implies adding a user database to the already existing system databases.

Sybase ASE 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.

To migrate the schemas from Oracle, databases are created in Sybase ASE for the schemas. The databases will hold the objects and their data.

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

7.3.1 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 Sybase ASE 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.

7.3.2 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 Oracle schema 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**. To mimic the Oracle schema, a login with the same name as the schema can be made the database owner.
- **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 Sybase ASE puts the database on one or more of the default device.
- **Transaction log.** Unlike Oracle, where rollback segments and redo logs are central to the instance, 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 Sybase ASE, 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**.

7.3.3 SEGMENTS:

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

When database is created, Sybase ASE 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 segment_name**

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

To configure the settings for an entire database, system administrator or database owner can use database options.

Segment names are used in **create table** and **create index** commands to place tables or indexes on specific database devices. Using segments can improve Sybase ASE performance and give the System Administrator or Database Owner increased control over the placement, size, and space usage of database objects.

7.3.4 PARTITION:

A *partition* is block of storage for a table. Partitioning a table splits it so that multiple tasks can access it simultaneously. When partitioned tables are placed on segments with a matching number of devices, each partition starts on a separate database device.

Partitioning is the basis for parallel processing, which can significantly improve performance. Sybase ASE supports horizontal partitioning, in which a selection of table rows can be distributed among disk devices. Individual table or index rows are assigned to a partition according to a partitioning strategy. There are several ways to partition a table:

- **Round-robin** – by random assignment.
- **Hash** – using a hashing function to determine row assignment (semantics-based partitioning).
- **Range** – according to values in the data row falling within a range of specified values (semantics-based partitioning).
- **List** – according to values in the data row matching specified values (semantics-based partitioning).

7.3.5 EXAMPLE:

This is an example of creating Sybase devices, database and segments. The test table and index created are in the same database but placed on two different logical devices.

```
disk init name = "test_data" physname = "/sybase/device_data"
go

disk init name = "test_index" physname = "/sybase/device_index"
go

disk init name = "test_log" physname = "/sybase/device_log"
go

create database testdb on test_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
```

7.3.6 IMPROVE PERFORMANCE WITH OBJECT PLACEMENT:

Sybase ASE 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.

8 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 Oracle database objects to Sybase database objects and changing the Oracle specific SQL code in the application code.
3. **Validation:** This subtask accounts for all the data validation and verifies data integrity.

8.1 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.
- Type of Data 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 CPU on the server.

8.2 OPTIONS FOR MIGRATION

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

8.2.1 POWERDESIGNER

Using the PowerDesigner tool you will be able to reverse engineer most of the Oracle database objects. These can then be stored into a PowerDesigner Physical Data Model (PDM). From this model you will be able to create the Sybase database object scripts. See Appendix G 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 a lot of exceptions the use of CIS gives a simpler mechanism to insert data from source to target tables.

8.2.2 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.

Sybase ASE 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 Oracle export and Sybase BCP utility, data from the oracle can be migrated to Sybase ASE via flat files:

- Retrieve the data from all the tables of from the Oracle database into a text file for each table with a field or column separator.
- Set following option in the 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 Oracle database, into corresponding ASE tables using bulk copying utility:
bcp <database>..

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

For more information about bulk copy, see Appendix H.

8.2.3 COMPONENT INTEGRATED SERVICES

CIS (Component Integration Services) provides transparent access to both Sybase and Oracle databases on different servers. Using CIS tables in remote servers can be accessed as 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 hold metadata. Internally, when a query involving remote tables is executed, the storage location is determined, and the remote location is accessed so that data can be retrieved. The datatypes of the oracle table are converted into the specified Sybase ASE types automatically when the data is retrieved.

8.2.4 ENTERPRISE CONNECT DATA ACCESS OPTION FOR ORACLE

In order to enable CIS access to Oracle tables you need to install the Sybase product Enterprise Connect Data Access Option for Oracle. ECDA Option for Oracle provides Open Client access to Oracle databases. It operates in conjunction with the Sybase ASE/Component Integration Services feature (ASE/CIS) and as a standalone gateway.

Using Sybase ASE, you can join Oracle tables with tables in Sybase ASE. Access to these objects through Sybase ASE is transparent to the application and allows you to copy data from Oracle tables to Sybase ASE tables. The main advantage is that ECDA takes care of the datatype conversions.

8.2.4.1 ECDA EXAMPLE

To allow access to Oracle tables via CIS, the Enterprise Connect Data Access Option for Oracle must be installed. In order to create a proxy table for an Oracle table the Oracle server must be accessible as a remote server from within Sybase ASE.

The ECDA is a separate process, like the backup process, and needs to be started separately.

Once the ECDA server is running, define the remote Oracle server:

```
sp_addserver ORACLEDC, direct_connect, ORACLEDC
```

In the Oracle database there is the following table:

```
example_table  
(id_num int,  
name varchar(30),  
phone varchar(20) null,  
birthdate date null)
```

In the Sybase ASE database the following proxy tables will be created:

```
create existing table example_table  
(id_num int,  
name varchar(30),  
phone varchar(20) null,  
birthdate smalldatetime null)  
external table  
at 'ORACLEDC.oradb..example_table'
```

Now you can access the example_table from isql as if the table is inside the Sybase ASE database.

This is just a short version of what needs to be happening to allow remote database access via CIS and ECDA. However, once configured and tweaked, these are the basic steps.

8.3 MIGRATING THE DATABASE OBJECTS AND SQL APPLICATION CODE

8.3.1 MIGRATING DATABASE OBJECTS

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

Oracle	Sybase ASE
Constraints	Constraints
Database links	Remote Server
Functions	Stored functions
Index	Index
Index-organized table	Table with clustered Index
Package	Stored procedure
Procedure	Stored procedure
Sequence	Identity
Synonym	Similar functionality with views for table and view synonyms. All other synonyme references must be replaced with fully qualified object strings.
Table	Table
Triggers	Triggers
View	View

8.3.1.1 CONSTRAINTS:

The functionality provided by Oracle and Sybase ASE to define constraints on columns are almost identical.

Following are the common constraint used by Oracle and Sybase ASE:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- REFERENCE KEY
- CHECK

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

8.3.1.2 TRIGGERS:

Oracle allows multiple triggers to be created on the same table. While Sybase allows only one trigger to be created on a table for update, delete and insert.

In Sybase ASE, 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 Sybase ASE detects a severe error, the entire transaction is rolled back.

8.3.1.3 VIEWS:

Various types of views available in Sybase ASE:

- 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

8.3.1.4 STORED PROCEDURES:

Oracle provides overloading of stored procedure. While Adaptive Server does not support overloading, such procedures will have to be recreated using unique names.

8.3.1.5 NO EQUIVALENT OBJECTS OF ORACLE IN SYBASE:

There is no exact equivalent object in Sybase ASE, for Oracle objects like synonyms, sequences and Database links. However Sybase ASE can provide similar functionalities:

- **Synonyms:** The solution for synonyms is to create views that refer the object with database name, owner and table name for field reference. Views can have their own permissions and Sybase ASE allows CRUD operations through views for table and view synonyms. All other references to synonyms must be replaced with fully qualified object names.
- **Sequences:** Sybase ASE does not have direct equivalent for Sequences, but a table can have a column that are defined with **identity** property, which simulates the behavior of sequences of Oracle. One of the advantage of sequences is independent objects is that they can be used to supply unique identifier across multiple tables. This property of Oracle sequences is not inherent in Sybase ASE's identity property. A table and a stored procedure can be created to simulate this functionality of oracle:

Oracle code:

```
CREATE SEQUENCE test_seq
MINVALUE 1
STARTWITH 1
INCREMENTED by 1
CACHE 20;

INSERT INTO m_table VALUES (test.seq.nextval,...);
```

Equivalent Sybase Code:

```
// create table
CREATE TABLE my_seq (seq int)
go

//initialize the sequence
INSERT INTO my_seq select 0
go

/*create stored procedure to get the incremented value and to set the
incremental value */
```

```
CREATE PROCEDURE get_seq (@seq int OUTPUT)
AS
UPDATE my_seq SET seq = seq+1
SELECT @seq = seq FROM my_seq
go

// execute the sp to get the next sequence number
DECLARE @seq int
EXEC get_seq @seq OUTPUT
INSERT INTO m_table VALUES (@seq,...)
go
```

- **Database Links**

A database link is a pointer in the local database that lets you access objects on a remote database. In Adaptive

Server, similar functionality can be obtained by adding remote server and remote login to the Sybase ASE by using system stored procedure:

- **sp_addserver** - for adding remote server
- **sp_addremotelogin** - for adding remote login information to Sybase ASE
- **sp_adduser** – On the remote server you must be added as a user to the appropriate database.

The remote object in Sybase ASE can be referred by –

`remote_server_name.database_name.database_owner.object_name`

This is the equivalent to the @dblink attribute in Oracle:

```
select * from database_schema.object_name@remote_server_name
```

8.3.2 MIGRATING THE SQL APPLICATION CODE

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

- Stored procedures

- Functions
- Triggers
- SQL queries

See Appendix G for the details on the usage of the PowerDesigner tool. Also, refer to appendix D to find out about the SQL language differences and appendix F on any Tuning and Performance that may be needed. Any Oracle hints (overwrites on what the Oracle RDBMS would use for solving the query without the hint) need to be removed from the SQL code.

8.4 VALIDATE THE DATA MIGRATION

8.4.1 VERIFY THE DATA TRANSFER

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

8.4.2 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 constraints in the database require thorough testing of application to verify the data integrity.

9 APPLICATION MIGRATION

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 5. This section describes the four different types of client database applications that need to be migrated from Oracle to Sybase ASE.

- Embedded SQL application
- ODBC client application
- JDBC client application
- Database-specific library application
- C Applications
- Oracle forms

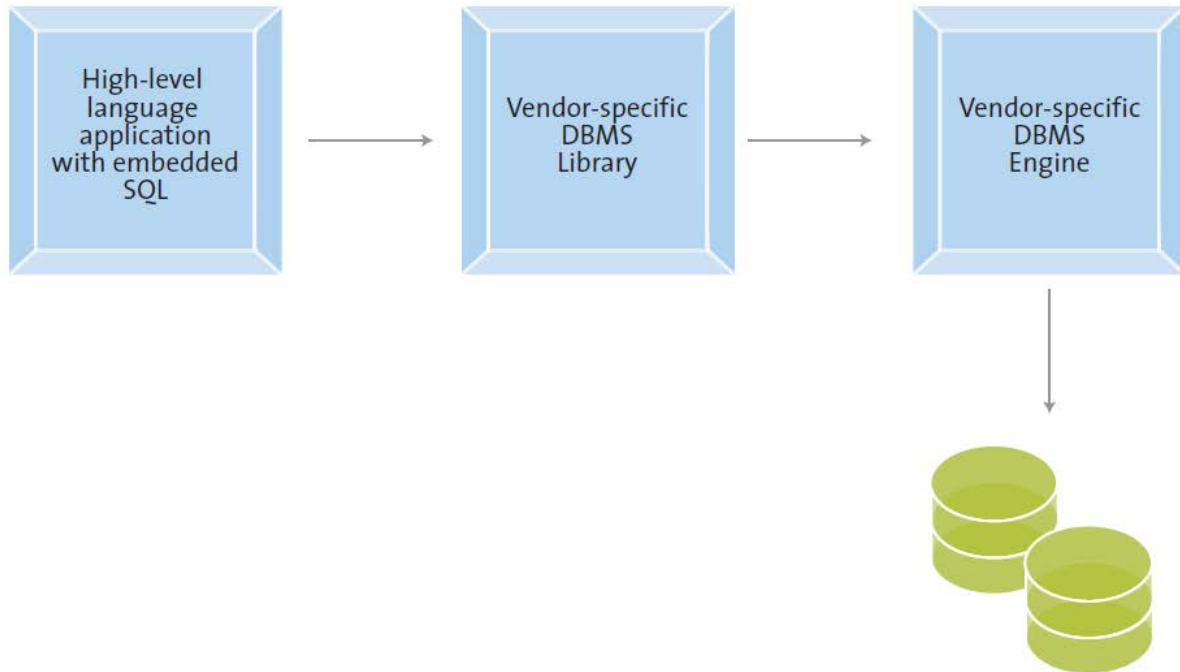
In all cases, conversion of one type of application to any of the other types of applications is possible. For example, instead of converting your Oracle Embedded SQL application to a Sybase Embedded SQL application, you can convert your Oracle Embedded SQL application to a JDBC client application.

Also, refer to appendix D to find out about the SQL language differences and appendix F on any Tuning and Performance that may be needed. Any Oracle hints (overwrites on what the Oracle RDBMS would use for solving the query without the hint) need to be removed from the SQL code

9.1 EMBEDDED SQL APPLICATION

Some Oracle Embedded SQL applications can be easily converted to Sybase Embedded SQL applications. Sybase ASE supports Embedded SQL for COBOL and C.

High-level languages that have embedded SQL in their code use a pre-compiler, which translates the embedded SQL code into Client-Library function calls.



The developer needs to check the Oracle Embedded SQL statements to see if the equivalent statement and arguments are supported by the target Sybase Embedded SQL statements.

The Oracle SQL string that is embedded in the application may need to be converted to a Sybase SQL string.

You can write Embedded SQL keywords in uppercase, lowercase, or mixed case.

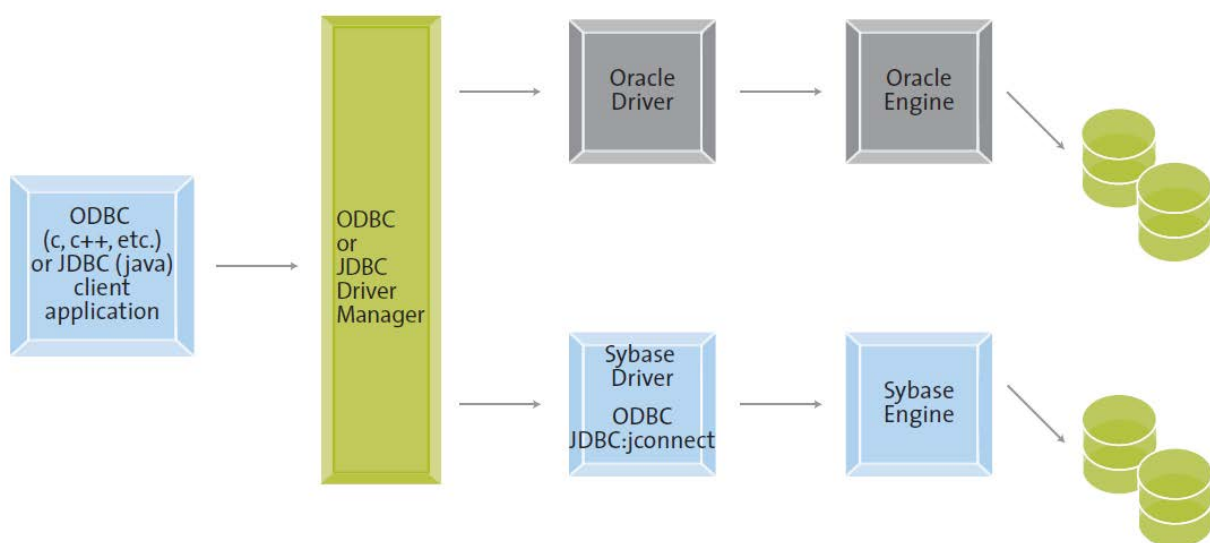
Following are the embedded functions:

Embedded SQL statement	Description
CLOSE	Closes an open cursor
COMMIT WORK Or COMMIT TRAN Or	Ends a transaction, preserving changes made to the database during the transaction

Embedded SQL statement	Description
COMMIT TRANSACTION	
ROLLBACK WORK Or ROLLBACK TRAN Or ROLLBACK TRANSACTION	Rolls a transaction back to a savepoint inside the transaction or to the beginning of the transaction
DESCRIBE INPUT	Obtains information about dynamic parameter markers in a prepared dynamic SQL statement and stores that information in a SQL descriptor
DESCRIBE OUTPUT	Obtains row format information about the result set of a prepared dynamic SQL statement
EXECUTE	Executes a dynamic SQL statement from a prepared statement
DECLARE CURSOR	Declares a cursor
OPEN CURSOR	open a cursor
EXECUTE IMMEDIATE	Executes a dynamic SQL statement stored in a characterstring host variable or quoted string
SQLCA SQLSTATE SQLCODE	These structures contain error, warning, and information message codes that data Server and Client-Library generate
FETCH	Copies data values from the current cursor row into host variables
CONNECT	To log on to a database
PREPARE	Declares a name for a dynamic SQL statement buffer

9.2 ODBC OR JDBC CLIENT APPLICATION

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



9.2.1 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 an Oracle 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 ASE
- Check the version of the ODBC client application and determine if all ODBC statements are supported.
- Configure the data source.

- 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 Sybase ASE with correct database.

- Possible changes in calling stored procedures
- Possible logical changes required to retain the program flow

9.2.2 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 ASE.
- To convert an Oracle 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.
- If the connectivity is obtained by using DataManager class, the change the connection statement that identifies the database driver that will be invoked.

For e.g., In order to connect from a Java application to an Oracle database using the OCI driver, you have to import the Oracle driver calls, register the driver manager, and connect with your user ID, the password, and the database name.

9.2.3 ORACLE JDBC CONNECTION

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;

// Load the driver
DriverManager.registerDriver(new
oracle.jdbc.driver.OracleDriver());

// Connect to the database
Connection conn =
DriverManager.getConnection
("jdbc:oracle:oci8:@oracle", "ui
d", "pwd");
```

9.2.4 SYBASE JDBC CONNECTION

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

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

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

DriverManager.registerDriver(sybDrive
r);

//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);
```

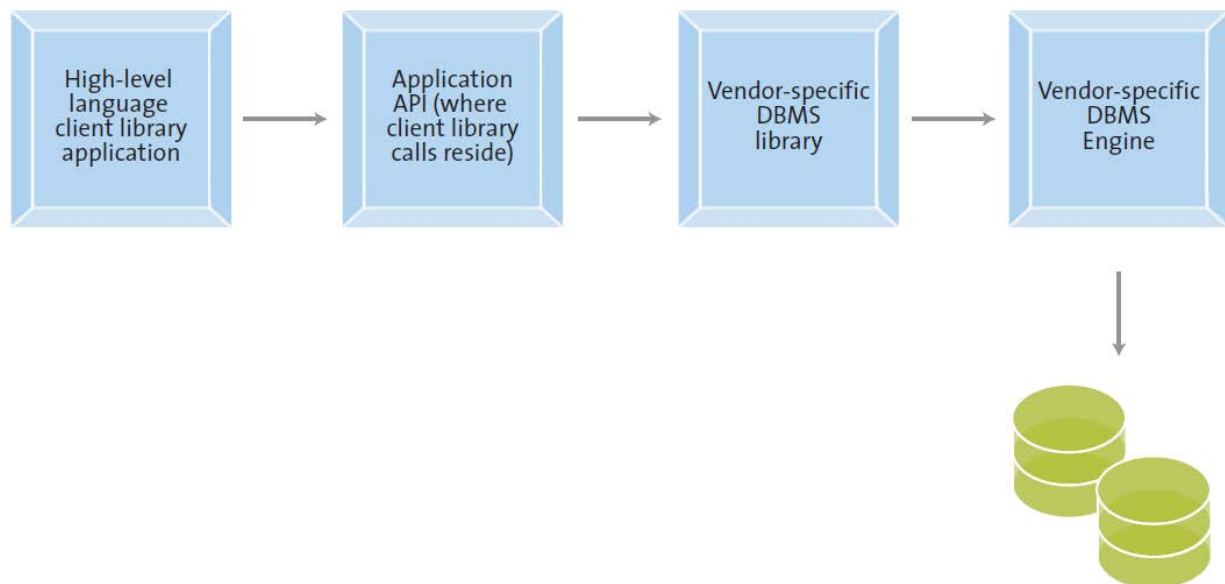
If the connectivity is obtained through a datasource, the application server should be properly configured.

If the SQL statements passed to the JDBC methods use any Oracle-specific syntax, then these items will need to be updated to reflect T-SQL syntax.

9.3 DATABASE-SPECIFIC CLIENT LIBRARY APPLICATION

Database-specific client library applications are the most difficult applications to migrate. If the original architecture of the client library application were designed modularly so that you could easily migrate to another database or work with more than one database, you would have defined your own application API. This way all of your changes would be localized to the Application API.

When using SQL*Net in the API it will be necessary to create a new API (based on CT-lib) to the application.



If there isn't a separation between the application and calls to the vendor-specific RDBMS API, then the developer's changes will be intermingled with the function of the application.

Similar to all other types of client applications, the SQL string that is executed must be compatible with the RDBMS' syntax.

9.4 C APPLICATIONS

While migrating client application, which uses C language, the developer needs to change all the calls of Oracle OCI to the appropriate equivalent calls of Sybase CT-lib library. Client-Library is a collection of routines for sending commands to and retrieving results from Sybase ASE servers.

In most cases, you can replace OCI functions with the appropriate Ct-Library functions, followed by relevant changes to the supporting program code. The remaining non-OCI program code should require minimal modification.

The program flow is retaining but you need to modify the processing of database handles. There may not be an exact match in the conversion process. Program might require addition revisions to obtain similar functionality.

The following examples show you the different SQL statements in order to connect to a database. In Oracle you need to define variables for the environment handles as well as the database name, username, and password:

```
OCILogon(    env_hp,          // environment handle
err_hp,      // error handle
&svc_hp,     // service context
user_name,   // username
strlen (user_name), // length of username
password,    // password
strlen (password), // length of password
db_name      // database name
strlen (db_name)); // length of database name
```

In Ct-Lib, you need allocate the connection structure first and then set the user and password to the connection by using `ct_con_props()` function. And then create the connection. So, the OCI statement will be converted as:

```
/* First, allocate a connection structure. */
ret = ct_con_alloc(context, &connection);

/* These two calls set the user credentials (username and
** password) for opening the connection. */
```

```
ret = ct_con_props(connection, CS_SET, CS_USERNAME,  
Ex_username, CS_NULLTERM, NULL);  
ret = ct_con_props(connection, CS_SET, CS_PASSWORD,  
Ex_password, CS_NULLTERM, NULL);  
  
/* Create the connection. */  
ret = ct_connect(connection, (CS_CHAR *) EX_SERVER, strlen(EX_SERVER));
```

Developer needs to set the environment variable for connecting to the Sybase ASE database. E.g. following environment variables needs to be set for CT-Library calls:

```
PATH=sybase_bin_path  
SYBASE=sybase_path  
LD_LIBRARY_PATH=sybase_lib_path  
DSQUERY=server_name
```

Some of functions of OCI have no equivalents in Ct-Library. The functionality must be implemented either in SQL or in the C application directly. For example, following functions of OCI:

```
OCI_SERVER_VERSION()
```

Transaction functions:

```
OCI_TRANS_COMMIT()
```

```
OCI_TRANS_ROLLBACK()
```

```
OCI_TRANS_PREPARE()
```

Navigational Functions:

```
OCI_CACHE_FLUSH()
```

```
OCI_CACHE_REFRESH()
```

Date Functions:

`OCIDateCheck()`

`OCIDateLastDate()`

`OCIDateNextDate()`

`OCIIntervalAdd()`

Table Functions:

`OCITableSize()`

`OCITableExists()`

9.4.1 ERROR HANDLING IN CT-LIBRARY

Ct-Library generates messages in response to a wide range of error and informational conditions. These messages are called “Client-Library messages” or “client messages.”

Servers also generate messages in response to error and informational conditions. These messages are called “server messages.”

Callbacks can be used in applications to handle Client-Library and server messages. The application should define and install callback routines to handle Client-Library and server messages. When a message is generated, Client-Library calls the appropriate callback and passes details about the message using the callback’s input parameters.

The mapping of most frequently used OCI calls to the closet Ct-Library equivalents:

Oracle OCI function	Sybase CT-Lib function
<i>To Set environment</i>	
OCIConnectionPoolCreate	cs_ctx_alloc
OCIEnvCreate	ct_init
	ct_config
	cs_condif
<i>For error Handling</i>	
OCIUserCallbackRegister	Cs_config(CS_MESSAGE_CB)
	Ct_callback
<i>Connect To Server</i>	
OCILogon	ct_con_alloc
OCISessionBegin	ct_con_props
OCIServerAttach	ct_connect
	ct_options
<i>Language Command</i>	
OCIStmtPrepare	Ct_cmd_alloc
OCIStmtExecute	Ct_command

Oracle OCI function	Sybase CT-Lib function
	Ct_send
To Process Result	
OCISmtFetch	Ct_results
OCIBindByName	Ct_fetch
OCIBindByPos	Ct_bind
OCIBindDymanic	Ct_describe
OCIBindObject	Ct_res_info
OCISmtGetBindInfo	
To Finish	
OCIConnectionPoolDestroy	Ct_cmd_drop
OCIBreak	Ct_cancel
OCITerminate	Ct_close
	Ct_exit
	Cs_ctx_drop
Conversion functions	
OCIDateFromText	Cs_convert
OCIDateToText	
OCIStrAssignText	
OCIDateTimeConvert	
For Date	
OCIDateGetDate	Cs_time
OCIDateGetTime	
For Date time compare	
OCIDateCompare	Cs_cmp
OCIDateTimeCompare	
Miscellaneous	

Oracle OCI function	Sybase CT-Lib function
OCIErrorGet	Cs_diag Ct_callback

9.5 ORACLE FORMS

Oracle Forms, a component of the Oracle Developer Suite, is Oracle's tool to design and build enterprise applications quickly and efficiently. Oracle Forms-based applications can retrieve and manipulate data in Oracle databases. These forms can also be deployed across the Web.

Oracle form is the basis for the user interface. It is graphical in nature and is used to present data and accept user input.

PowerBuilder® is an enterprise development tool that allows you to build many types of applications and components. It is one of a group of Sybase products that together provide the tools to develop client/server, multi-tier, and Internet applications.

A PowerBuilder application can contain:

- **A user interface** – Menus, windows, and window controls that users interact with to direct an application.
- **Application processing logic** – Event and function scripts in which you code business rules, validation rules, and other application processing. PowerBuilder allows you to code application processing logic as part of the user interface or in separate modules called custom class user objects.

Oracle Forms applications can be rewritten using PowerBuilder. Most of the functionality provided by Oracle forms can be retrieved by using PowerBuilder with Sybase ASE.

9.5.1 MIGRATION APPROACH

Migration of Oracle Forms application to PowerBuilder application is not straightforward. The form is the basis of user interface (UI) in Oracle while windows is basis of UI in PowerBuilder. Both are graphical in nature and are used to present data and accept user input. Both can contain elements graphical and non-graphical in nature.

Like other GUI applications, both Oracle Forms application and PowerBuilder application are composed of two parts: the application logic and the user interface. Oracle Forms maintains this separation throughout its design. Hence, the Forms Runtime is the same in both the two-tier and three-tier architectures.

9.5.2 EVALUATING ORACLE FORMS

Oracle Forms applications are Oracle database optimized transactional applications built using the tools provided by Oracle's Developer Suite. Oracle Forms Builder, a part of Oracle Developer suite, enables you to quickly and easily create database forms and business logic. The development is done using wizards, drag-and-drop components, and built-in functions. A visual editor can be used to refine the forms with graphics and other components. The coding language is PL/SQL. A client/server Forms application can run on the Web with very little or no modification.

An Oracle Forms application is an organization of items. The organization of the items has two perspectives: that of the developer, called the form view; and that of the user, called the visual (or presentation) view. In the form view, the items are organized into logical blocks and have no visual structure. The visual structure is provided by constructing the visual view where items are placed on a canvas and presented to the user in windows. It is very important to maintain the same visual view when an Oracle Forms application is migrated. The PowerBuilder developers need to look at the visual view to understand how many screen layouts they need in PowerBuilder. Each item in a canvas will correspond to one window in PowerBuilder.

The form view is used by the developer to access the business logic and the data access. The visual view is used to build the user interface.

An Oracle Forms application is built using components called modules. The following four types of modules are visible when a form application is opened up in Oracle Forms Builder:

1. **Object Library** – The object-oriented programming concept of reusing code is implemented in Oracle Forms by the object library and PL/SQL library modules. The Object library contains reusable objects and the PL/SQL library contains reusable code. Hence, these are examined first for a migration.
2. **PL/SQL Library** – The PL/SQL library contains reusable code invoked by other form, menu, or library modules.
3. **Form Module** – A form (or form module) is the main component that anchors an application and provides the necessary code to interact with the datasource and the user interface. The underlying database data is reflected in multiple items, including fields, check boxes, and radio groups. A form is logically organized into blocks.

- Data Block: serves as a bridge to the underlying data and provides an abstraction for how the data is reached.
 - Control Blocks: The control blocks are more like programming units that can organize controls into logical groups forming part of the same user interface navigation cycle.
4. **Menu Module** – The menu module consists of a hierarchy of menus. Each menu consists of the items that can be selected. Menu modules are usually attached to form modules. Every form module has a default menu that includes the commands for all basic database operations, such as insert, update, delete, and query.

So, Oracle form application can be built using multiple forms, menus, and library modules. The Oracle Forms Builder can be used to browse the component model and understand the modules and their characteristics so that the Forms Model can be built. The elements offered by these modules account for the capabilities, features, and functioning of the application and should be the focus of the developer involved in migrating an application.

9.5.3 UNDERSTANDING POWERBUILDER

In PowerBuilder, you work with one or more targets in a workspace. You can add as many targets to the workspace as you want, open and edit objects in multiple targets, and build and deploy multiple targets at once.

A PowerBuilder target can be one of two types:

- **PowerScript target** – A client/server or multi-tier executable application or a server component.
- **Web target** – A Web target that can contain all the elements you need to build a Web site application—HTML files, scripts, images, downloaded components—or a JavaServer Pages (JSP) application. A Web target also contains settings for build options, database connections, and deployment.

The basic building blocks of a PowerScript target are objects:

- **Application Object:** The Application object is the entry point into an application. It is a discrete object that is saved in a PowerBuilder library (PBL file), just like a window, menu, function, or DataWindow® object.

The Application object defines application-level behavior, such as which fonts are used by default for text, and what processing should occur when the application begins and ends.

When a user runs the application, an Open event is triggered in the Application object. The script you write for the Open event initiates the activity in the application. When the user ends the application, the Close event in the Application object is triggered.

- **Windows:** Windows are the primary interface between the user and a PowerBuilder application. Windows can display information, request information from a user, and respond to the user's mouse or keyboard actions.

A window consists of:

- Properties that define the window's appearance and behavior (for example, a window might have a title bar and a Minimize box)
- Events triggered by user actions
- Controls placed in the window:

You place controls in a window to request and receive information from the user and to present information to the user. All window controls have events so that users can act on the controls. You write scripts that determine the processing that takes place when an event occurs in the control.

By adding customized menus and toolbars to your applications, you can make it easy and intuitive for your users to select commands and options.

- **DataWindow objects:** A DataWindow object is an object that you use to retrieve, present and manipulate data from the database. Using DataWindow, you can display the data in the format that best presents the data to your users. You can specify a simple validation rule for the data, without writing any code, to make sure users enter valid data. Also, you can include computed fields, pictures, and graphs that are tied directly to the data retrieved by the object.
- **Menus:** Menus are lists of items that a user can select from a menu bar for the active window.
- **Queries:** A query is a SQL statement that is saved with a name so that it can be used repeatedly as the data source for a DataWindow object. Queries enhance developer productivity, because they can be coded once but reused as often as necessary.
- **User Objects:** If you find yourself using the same application feature repeatedly, you should define a user object. You define the user object once and use it as many times as you need. User objects can be visual or class (non-visual).
 - **Visual user objects** These are reusable controls or sets of controls that have a consistent behavior. For example, a visual user object could consist of several buttons that function as a unit. The buttons could have scripts associated with them that perform standard processing. Once the object is defined, you can use it as often as you need.
 - **Class user objects** These are reusable processing modules that have no visual component. You typically use class objects to define business rules and other processing that acts as a unit. For example, you might want to calculate commissions or perform statistical analysis in several applications. To do this, you could define a class user object. To use a class user object, you create an instance of the object in a script and call its functions.
- **Libraries:** You save objects, such as windows and menus, in PowerBuilder libraries (PBL files).

When you run an application, PowerBuilder retrieves the objects from the library. Applications can use as many libraries as you want. When you create an application, you specify which libraries it uses.

- **Projects:** You can create Project objects that build executable applications and components you can deploy to a server.

In PowerBuilder, you edit objects such as applications, windows, menus, DataWindow objects, and user objects in painters. In addition to painters that edit objects, other painters such as the Library painter and the Database painter provide you with the ability to work with libraries and databases.

PowerScript applications are event-driven:

Users control the flow of the application by the actions they take. When a user clicks a button, chooses an item from a menu, or enters data into a text box, an event is triggered. You write scripts that specify the processing that should happen when the event is triggered.

Document and redesign the Oracle Form application to build the new PowerDesigner application for Adaptive Enterprise Server. For more information about PowerBuilder, see the PowerBuilder's User Guide.

9.5.4 VALIDATION OF POWERBUILDER APPLICATION

The PowerBuilder application is newly written application. None of the Oracle forms code is reusable. Hence, the application should undergo through unit testing.

The objective of unit testing is to verify that the user interface closely resembles the Oracle Forms application and all the components function correctly. Every Frame of the Oracle Form can be compared with corresponding window of the PowerBuilder.

Check the event model of PowerBuilder correctly implemented as in Oracle Forms. Verify it by performing walkthrough of the entire application by running it in debug mode.

In debug mode of PowerBuilder, you can insert breakpoints (stops) in scripts and functions, single-step through code, and display the contents of variables to locate logic errors that will result in errors at runtime.

The application should be tested for functionality. Start with testing the menus and then drilling down window-by-window. Test the functionality of every control. Use same set of testcases in both the applications. Verify that all the functional areas and navigation match the old application.

9.6 TRIGGERS AND STORED PROCEDURES

This chapter dives into the differences between Oracle and Sybase ASE on the trigger and stored procedure level. It is contained in the application migration section, because triggers and stored procedures contain code or at least contain some logic, like enforcing referential integrity for an application.

9.6.1 TRIGGERS

Oracle triggers are either BEFORE triggers or AFTER triggers. BEFORE triggers are used when the trigger action determines whether the triggering statement should be allowed to complete. Oracle uses BEFORE trigger to avoid unnecessary processing of the triggering statement and its eventual rollback in cases where an exception is raised.

As combinations, there are four different types of triggers in Oracle:

- BEFORE STATEMENT Trigger
- BEFORE ROW Trigger
- AFTER STATEMENT Trigger
- AFTER ROW Trigger



Sybase ASE only supports AFTER triggers. But unlike Oracle, Sybase ASE maintains the concept of 2 shadow tables that are maintained during the life of a trigger.

These tables are called:

- **Inserted:** This table contains all the new, not yet committed, data that the transaction is about to add to the table on which this trigger has been executed.
- **Deleted:** This table contains all the data that currently exists, fully committed, for this transaction.

You can simply say: inserted contains the new and deleted contains the old. These two tables can be joined to each other or to other accessible tables. There is virtually no limit on the combinations.

9.6.2 STORED PROCEDURES

Oracle has three different kinds of stored procedures, namely functions, stored procedures and packages. Oracle uses the PL/SQL language to describe the action within a stored procedure. The PL stands for procedural language, a string of if-then-else combinations. There are considerable differences between Oracle's PL/SQL and Sybase ASE Transact SQL (T/SQL). More information about these differences are described in the sections: PL/SQL vs. T/SQL Constructs and PL/SQL and T/SQL Language Elements later in this chapter.

9.6.2.1 INDIVIDUAL SQL STATEMENTS

Each individual SQL statement has to be examined and analyzed for compatibility. In most cases there will be little or no changes necessary, but there are several SQL constructs in Oracle that are not possible in Sybase ASE. To streamline this activity it is highly recommended to use automatic conversion utilities to perform these conversions.

See Appendix G for an example of an automatic conversion utility.

9.6.2.2 LOGICAL TRANSACTION HANDLING

In Oracle, transactions are implicit as set by the ANSI standard. The implicit transaction model requires that each SQL statement is part of a logical transaction. A new logical transaction is automatically initiated when a COMMIT or ROLLBACK command is executed. This also implies that data changes from an individual SQL statement are not committed to the database after execution. The changes are committed to the database only when a COMMIT statement is run.

In Sybase ASE, transactions are explicit by definition. This implies that an individual SQL statement is not part of a logical transaction by default. A SQL statement belongs to a logical transaction if the transaction explicitly initiated by a user with a BEGIN TRANSACTION (or BEGIN TRAN) statement is still in effect. The logical transaction ends with a corresponding COMMIT TRANSACTION (or COMMIT TRAN) or ROLLBACK TRANSACTION (or ROLLBACK TRAN) statement. Each SQL statement that is not part of a logical transaction is committed on completion.



Before you go ahead and add a BEGIN TRAN statement at the beginning of every session and after every COMMIT or ROLLBACK statement to create the logical transactions, you need to be aware of the consequences by doing so. While keeping in mind the Sybase ASE does create a shared lock on every page and row for SELECT statement for the duration of a transaction, you quickly realize that you will create an unsustainable blocking situation in a multi user environment. Or even worst, a series of deadlocks and deadlock victims due to excessive long running transactions.

The better way is to analyze the transactions and determine which transactions are Ok to keep intact and which transactions need to be split in smaller, shorter transactions. There are 3rd party tools available that help you in identifying these logical transactions.

See Appendix G for an example of a logical transaction analysis tool.

9.6.2.3 ERROR HANDLING WITHIN THE STORED PROCEDURE

Oracle PL/SQL checks each SQL statement for errors before proceeding with the next statement. If an error occurs, control immediately jumps to an exception handler. This avoids you having to check the status of every SQL statement.

Sybase ASE does not check for errors after each T/SQL statement. Control is passed to the next statement, irrespective of the error conditions created by the previous statement. It is your responsibility to check for errors after the execution of each SQL statement. Failure to do so may result in erroneous results.

Here are some examples on handling error exceptions:

Oracle	Sybase ASE
<pre> DECLARE salary_too_high EXCEPTION; current_salary NUMBER := 20000; max_salary NUMBER := 10000; erroneous_salary NUMBER; BEGIN BEGIN ----- sub-block begins IF current_salary > max_salary THEN RAISE salary_too_high; -- raise the exception END IF; EXCEPTION </pre>	<pre> begin declare @current_salary int, @max_salary int, @error_txt varchar(1024) set @current_salary=20000, @max_salary=10000 if @current_salary > @max_salary begin select @error_txt='Salary ' + convert(varchar(20), @current_salary) + ' </pre>

```
    WHEN salary_too_high THEN
        -- first step in handling the error
        DBMS_OUTPUT.PUT_LINE('Salary ' ||
            erroneous_salary || ' is out of range.');
```

```
        DBMS_OUTPUT.PUT_LINE
            ('Maximum salary is ' || max_salary ||
                ' ');
        RAISE; -- reraise the current exception
    END; ----- sub-block ends
EXCEPTION
    WHEN salary_too_high THEN
        -- handle the error more thoroughly
        erroneous_salary := current_salary;
        current_salary := max_salary;
        DBMS_OUTPUT.PUT_LINE('Revising salary from
            ' || erroneous_salary ||
                ' to ' || current_salary || ' ');
    END;
/
```

```
        is out of range.'+'Maximum salary is ' +
        convert(vchar(20),@max_salary) + '.'
            raiserror 99999 @error_txt
        end
    end
```

9.6.3 DATA TYPES

9.6.3.1 LOCAL VARIABLE



PL/SQL local variables can also be either of the following composite data types allowed by PL/SQL:

- RECORD These special PL/SQL variables cannot be translated into T-SQL
- TABLE an alternative approach must be developed.

T/SQL local variables can be any server data type except TEXT and IMAGE. PL/SQL local variables can be any server data type including the following:

- BINARY_INTEGER
- BOOLEAN

9.6.3.2 SERVER DATA TYPE

Please see Appendix C for a complete list of data types and their corresponding counterparts.

9.6.3.3 COMPOSITE DATA TYPE

Unlike Oracle, Sybase ASE does not support composite data types. Therefore all references of composite data types in Oracle PL/SQL code must be identified and rewritten to conform to T/SQL standards.

Oracle composite data types are defined as follows:

RECORD	You can declare a variable to be of type RECORD. Records have uniquely named fields. Logically related data that is dissimilar in type can be held together in a record as a logical unit.
TABLE	PL/SQL tables can have one column and a primary key, neither of which can be named. The column can belong to any scalar data type. The primary key must belong to type BINARY_INTEGER.



Both datatypes can be implemented in Sybase ASE by using temporary tables. There is some significant coding effort involved to correctly translate this scenario.

9.6.4 SCHEMA OBJECTS

Each schema object is compared in separate tables on create, drop, execute and alter, where applicable. The tables are divided into the following sections:

- Syntax
- Description
- Permissions
- Examples

Some tables are followed by a recommendations section that contains important information about conversion implications.

9.6.4.1 PROCEDURES

9.6.4.1.1 CREATE PROCEDURE

Oracle	Sybase ASE
<pre>CREATE or REPLACE PROCEDURE procl (a IN number, b OUT number, c IN OUT number) AS BEGIN . . . END</pre>	<pre>CREATE PROCEDURE procl @a int, @b int out, c int out AS BEGIN . . . END</pre>
<p>Description:</p> <p>The OR REPLACE keywords replace the procedure by the new definition if it already exists.</p> <p>The parameters passed to the PL/SQL procedure can be specified as IN (input), OUT (output only), or IN OUT (input and output). In the absence of these keywords, the parameter is assumed to be the "IN" parameter.</p> <p>The keyword IS or AS indicates the start of the procedure. The local variables are declared after the keyword IS or AS and before the keyword BEGIN.</p> <p>The BEGIN and END keywords enclose the body of the procedure.</p>	<p>Description:</p> <p>The CREATE PROCEDURE statement creates the named stored procedure in the database.</p> <p>You can optionally specify the parameters passed to the procedure as OUTPUT. Values of OUTPUT variables are available to the calling routine after the procedure is executed. The parameters specified without the OUTPUT keyword are considered as input parameters.</p> <p>The keyword AS indicates the start of the body of the procedure. The BEGIN and END keywords that enclose the stored procedure body are optional; all the procedural statements contained in the file after AS are considered part of the stored procedure if BEGIN and END are not used to mark blocks.</p>

Oracle

Sybase ASE

Alternatively you can use the following construct to define the body of the procedure:

- **external name**

creates an extended stored procedure.

- **dll_name**

specifies the name of the dynamic link library (DLL) or shared library containing the functions that implement the extended stored procedure. The dll_name can be specified with no extension or with a platform-specific extension, such as .dll on Windows NT or .so on Sun Solaris.

See the T/SQL and PL/SQL Language Elements section of this chapter for more information about the constructs allowed in T/SQL procedures.

Permission:

To create a procedure in your own schema, you must have the CREATE PROCEDURE system privilege. To create a procedure in another user's schema, you must have the CREATE ANY PROCEDURE system privilege.

Permission:

You must have the CREATE PROCEDURE system privilege to create the stored procedures.

9.6.4.1.2 RECOMMENDATIONS



Functional identical parts can be identified in the PL/SQL procedure and the T/SQL procedure structure. Therefore you can automate the conversion of most of the constructs from Oracle to Sybase ASE. In Sybase ASE each procedure must be dropped explicitly before replacing it.

See Appendix G for an example of a procedure conversion automation tool.


9.6.4.1.3 EXECUTE PROCEDURE

Oracle	Sybase ASE
<pre>VAR a NUMBER; VAR b NUMBER; VAR c NUMBER; exec procl (:a, :b , :c)</pre>	<pre>Declare @a int, @b int, @c int, @return_status exec @return_status = procl @a, @b, @c</pre>
<p>Description:</p> <p>Oracle PL/SQL procedures send data back to the calling routine by means of OUT parameters. Oracle offers functions that are a different type of schema objects. Functions can return an atomic value to the calling routine using the RETURN statement. The RETURN statement can return value of any data type.</p> <p>The formal_parameter is the parameter in the procedure definition. The actual_parameter is defined in the local block which calls the procedure supplying the value of the actual parameter for the respective formal parameter. The association between an actual parameter and formal parameter can be indicated using either positional or named notation.</p>	<p>Description:</p> <p>Sybase ASE stored procedures can only return integer values to the calling routine using the RETURN statement. In the absence of a RETURN statement, the stored procedure still returns a return status to the calling routine. This value can be captured in the "return_status" variable.</p> <p>The formal_parameter is the parameter in the procedure definition. The actual_parameter is defined in the local block which calls the procedure supplying the value of the actual parameter for the respective formal parameter. The association between an actual parameter and formal parameter can be indicated using either positional or named notation.</p> <p>Sybase ASE maintains a special parameter:</p> <ul style="list-style-type: none"> • with recompile <p>forces compilation of a new plan. Use this option if the parameter you are supplying is atypical or if the data has significantly changed. The changed plan is used on subsequent executions. Sybase ASE ignores this option when executing an extended system procedure.</p> <p>Using execute procedure with recompile many times can adversely affect the procedure cache performance. Since a new plan is generated every time you use with recompile, a useful performance plan may be pushed out of the cache if there is insufficient space for new plans.</p>

Oracle	Sybase ASE
<p>Positional notation:</p> <p>The actual parameters are supplied to the procedure in the same order as the formal parameters in the procedure definition.</p>	<p>Positional notation:</p> <p>The actual parameters are supplied to the procedure in the same order as the formal parameters in the procedure definition.</p>
<p>Named notation:</p> <p>The actual parameters are supplied to the procedure in an order different than that of the formal parameters in the procedure definition by using the name of the formal parameter as:</p> <pre>formal_parameter => actual_parameter</pre> <p>A constant literal can be specified in the place of the following:</p> <pre>as: formal_parameter => 10</pre> <p>If the formal_parameter is specified as OUT or IN OUT in the procedure definition, the value is made available to the calling routine after the execution of the procedure.</p>	<p>Named notation:</p> <p>The actual parameters are supplied to the procedure in an order different than that of the formal parameters in the procedure definition by using the name of the formal parameter as:</p> <pre>@formal_parameter = @actual_parameter</pre> <p>A constant literal can be specified in the place of the following:</p> <pre>'@actual_parameter ' as: @formal_parameter = 10</pre> <p>The keyword OUTPUT should be specified if the procedure has to return the value of that parameter to the calling routine as OUTPUT.</p>
<p>Permission:</p> <p>The user should have the EXECUTE privilege on the named procedure. The user need not have explicit privileges to access the underlying objects referred to within the PL/SQL procedure.</p>	<p>Permission:</p> <p>The user should have the EXECUTE permission on the stored procedure. The user need not have explicit privileges to access the underlying objects referred to within the stored procedure.</p>
<p>Example:</p> <p>Positional notation:</p> <pre>GetEmplName (EmpID); GetAllDeptCodes (:status); UpdateEmpSalary (EmpID,EmpName,:status); UpdateEmpSalary (13000,'Joe Richards');</pre> <p>Named notation:</p> <pre>UpdateEmpSalary (Employee => EmpName, Employee_Id => EmpID);</pre>	<p>Example:</p> <p>Positional notation:</p> <pre>EXEC GetEmplName @EmpID EXEC @status = GetAllDeptCodes EXEC @status = UpdateEmpSalary @EmpID,@EmpName EXEC UpdateEmpSalary 13000,'Joe Richards'</pre> <p>Named notation:</p> <pre>EXEC UpdateEmpSalary @Employee = @EmpName,</pre>

Oracle	Sybase ASE
<p>Mixed notation (where positional notation must precede named notation):</p> <pre>UpdateEmpSalary (EmpName, Employee_Id => EmpID); UpdateEmpSalary (Employee => EmpName, EmpID);</pre>	<pre>@Employee_Id = @EmpID</pre> <p>Mixed notation:</p> <pre>EXEC UpdateEmpSalary @EmpName, @Employee_Id = @EmpID EXEC UpdateEmpSalary @Employee = @EmpName, @EmpID</pre>

9.6.4.1.4 ALTER PROCEDURE

 Sybase ASE does not support the alter procedure command. In order to migrate alter procedure commands from Oracle, each individual alter procedure command has to be broken up into a drop procedure and a create procedure command.

9.6.4.2 FUNCTIONS

9.6.4.2.1 CREATE FUNCTION

Oracle	Sybase ASE
<p>Syntax:</p> <pre>CREATE OR REPLACE FUNCTION func1 (a NUMBER, b OUT NUMBER, c IN OUT NUMBER) RETURN VARCHAR2 AS BEGIN END</pre>	<p>Syntax:</p> <pre>create function func1 (@a int, @b int, @c int) returns varchar as begin . . . end</pre>
<p>Description:</p> <p>The OR REPLACE keywords replace the function with the new definition if it already exists.</p> <p>Parameters passed to the PL/SQL function can be specified as "IN" (input), "OUT" (output), or "IN OUT"</p>	<p>Description:</p> <p>The CREATE FUNCTION statement creates user-defined function in the database.</p> <p>All parameters are by default input parameters. Return values are defined as follows:</p>

Oracle

(input and output). In the absence of these keywords the parameter is assumed to be IN.

RETURN data type specifies the data type of the function's return value. The data type can be any data type supported by PL/SQL.

Sybase ASE

- **return_datatype**

is the return value of a scalar, user-defined function. It can be any of the scalar data types and Java ADTs except text, image, unitext and timestamp.

- **scalar_expression**

specifies the scalar value the scalar function returns. You can invoke scalar-valued functions where scalar expressions are used, including computed columns and check constraint definitions.

You can include these elements in a scalar function:

- declare statements to define data variables and cursors that are local to the function.
- Assigned values to objects local to the function (for example, assigning values to scalar and variables local to a table with select or set commands).
- Cursor operations that reference local cursors that are declared, opened, closed, and deallocated in the function.
- Control-of-flow statements.
- set options (only valid in the scope of the function).

Sybase ASE does not allow fetch statements in a scalar function that return data to the client. You **cannot** include :

- select or fetch statements that returns data to the client.
 - insert, update, or delete statements.
 - Utility commands, such as dbcc, dump and
-

Oracle

Sybase ASE

load commands.

- print statements
- Statement that references rand, rand2, getdate, or newid.

You can include select or fetch statements that assign values only to local variable.

The keyword AS indicates the start of the body of the procedure. The BEGIN and END keywords that enclose the stored procedure body are optional; all the procedural statements contained in the file after AS are considered part of the stored procedure if BEGIN and END are not used to mark blocks.

Permission:

To create a function in your own schema, you must have the CREATE PROCEDURE system privilege. To create a function in another user's schema, you must have the CREATE ANY PROCEDURE system privilege.

Permission:

Create function permission defaults to the database owner, who can transfer it to other users.

Owners of functions have execute permission on their functions. Other users do not have execute permissions unless execute permissions on the specific function are granted to them.

Examples:

```
CREATE FUNCTION get_bal
(acc_no IN NUMBER)
RETURN NUMBER
IS
acc_bal NUMBER(11,12);
BEGIN
SELECT balance
INTO acc_bal
FROM accounts
WHERE account_id = acc_no;
RETURN(acc_bal);
END;
```

Examples:

```
create function get_bal (@acc_no int)
returns numeric(11,12)
as
begin
declare @acc_bal numeric(11,12)
SELECT @acc_bal = balance
FROM accounts
WHERE account_id = acc_no;
return @acc_bal
end
```

9.6.4.2.2 RECOMMENDATIONS

Functional identical parts can be identified in the PL/SQL procedure and the T/SQL procedure structure. Therefore you can automate the conversion of most of the constructs from Oracle to Sybase ASE.



Sybase ASE supports 'scalar-valued SQL user-defined functions', which are functions returning a single value only. This is to distinguish from 'table-valued SQL user-defined functions', which return a table: Unlike Oracle, Sybase ASE does not support such user-defined functions at this point. If the Oracle function returns a table, an alternative solution must be found.

In Sybase ASE each function must be dropped explicitly before replacing it.

9.6.4.2.3 EXECUTE FUNCTION

Oracle	Sybase ASE
Description: Functions can return an atomic value to the calling routine using the RETURN statement. A function can be called as part of an expression.	Description: Functions can return an scalar value to the calling routine using the scalar_expression return statement. A function can be called as part of an expression.
Permission: You should have the EXECUTE privilege on the function to execute the named function. You need not have explicit privileges to access the underlying objects that are referred to within the PL/SQL function.	Permission: The user should have the EXECUTE permission on the function. The user need not have explicit privileges to access the underlying objects referred to within the function.
Example: <pre> 1) IF sp_str_replace2('a b c d e f', ' ', '- ',2, 1) THEN END IF; 2) var_1:=sp_str_replace2('a b c d e f', ' ', '- ',2, 1); </pre>	Example: <pre> select sp_str_replace2('a b c d e f', ' ', '- ', 2, 1) </pre> <p>where sp_str_replace2 returns this value: a b-c d e f</p>

Oracle

Sybase ASE

9.6.4.2.4 ALTER FUNCTION



Sybase ASE does not support the alter function command. In order to migrate alter function commands from Oracle, each individual alter function command has to be broken up into a drop function and a create function command.

9.6.4.3 PACKAGE AND PACKAGE BODY

Oracle allows to bundle all stored procedure and function definitions to be packaged together to share global variables, constants and cursors. Especially if you are working with the REF CURSOR type as return variable from functions and want to use the reference in another procedure, a package will enable you to do this.

The Oracle package body contains the PL/SQL code of the procedure or function. Dropping and recreating will not invalidate external references to stored procedures and functions. Oracle packages and package bodies are a widely used technique and provide powerful and easily deployable application packages.



Sybase ASE does not support packages of stored procedures or functions. To migrate Oracle packages and package bodies to Sybase ASE, every individual stored procedure or function has to be extracted from the package and the package body and reassembled to a create procedure or create function statement.

9.6.5 PL/SQL vs. T/SQL CONSTRUCTS

This section provides information about the Oracle constructs and equivalent Sybase ASE constructs.

9.6.5.1 CREATE PROCEDURE STATEMENT

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE procl (a OUT NUMBER) AS BEGIN a:=0; END;</pre>	<pre>DROP PROC procl GO CREATE PROC procl AS RETURN 0 GO</pre>

9.6.5.1.1 COMMENTS:

Translate create or replace into a DROP PROC followed by a CREATE PROC statement.

This Oracle function can be translated into a Sybase ASE procedure because of the integer return value. All Oracle stored procedures translate into Sybase ASE stored procedures.

9.6.5.2 PARAMETERS

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE procl (v_x IN NUMBER DEFAULT -1, v_y IN NUMBER, v_z OUT NUMBER, v_a IN CHAR DEFAULT 'TEST') AS BEGIN v_z:=0; END;</pre>	<pre>DROP PROC sp_procl GO CREATE PROC sp_procl @x int=-1, @y int, @z int OUT, @a char(20) = 'TEST' AS BEGIN SELECT @z=0 RETURN 0 END GO</pre>

9.6.5.2.1 COMMENTS:

Translate create or replace into a DROP PROC followed by a CREATE PROC statement.

Parameter passing is almost the same Oracle and Sybase ASE. By default, all the parameters are INPUT parameters, if not specified otherwise.



The @ sign in a parameter name declaration must be added in Sybase ASE to all the Oracle parameters as well as the length/size must be added to the data type definition.

9.6.5.3 DECLARE STATEMENT

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE proc1 AS v_x NUMBER(10,0); v_y NUMBER(19,2); v_z NUMBER(1,0); v_a CHAR(20); BEGIN RETURN; END;</pre>	<pre>DROP PROC sp_proc1 GO CREATE PROC sp_proc1 AS BEGIN DECLARE @x int, @y money, @z bit, @a char(20) RETURN 0 END GO</pre>

9.6.5.3.1 COMMENTS:

Oracle and Sybase ASE follow similar rules for declaring local variables.

9.6.5.4 IF STATEMENT

Oracle	Sybase ASE
<p>Example 1:</p> <pre>CREATE OR REPLACE PROCEDURE proc1 (v_Flag IN NUMBER DEFAULT 0) AS v_x NUMBER(10,0); BEGIN IF (v_Flag = 0) THEN v_x := -1; ELSE v_x := 10; END IF END;</pre>	<p>Example 1:</p> <pre>DROP PROC proc1 GO CREATE PROC proc1 @Flag int = 0 AS BEGIN DECLARE @x int IF (@Flag=0) SELECT @x = -1 ELSE SELECT @x = 10 END</pre>

Oracle	Sybase ASE
	GO
Example 2:	Example 2:
<pre>CREATE OR REPLACE PROCEDURE procl (v_Flag IN CHAR DEFAULT '') AS v_x NUMBER(10,0); BEGIN IF (v_Flag = '') THEN v_x := -1; ELSE IF (v_Flag = 'a') THEN v_x := 10; ELSE IF (v_Flag = 'b') THEN v_x := 20; END IF; END IF; END IF; END;</pre>	<pre>DROP PROC procl GO CREATE PROC procl @Flag char(2) = '' AS BEGIN DECLARE @x int IF (@Flag='') SELECT @x = -1 ELSE IF (@Flag = 'a') SELECT @x = 10 ELSE IF (@Flag = 'b') SELECT @x = 20 END GO</pre>
Example 3:	Example 3:
<pre>CREATE OR REPLACE PROCEDURE procl AS v_x NUMBER(10,0); v_temp NUMBER(1, 0) := 0; BEGIN SELECT 1 INTO v_temp FROM DUAL WHERE EXISTS (SELECT * FROM table2); IF v_temp = 1 THEN v_x := -1; END IF; END;</pre>	<pre>DROP PROC procl GO CREATE PROC procl AS BEGIN DECLARE @x int IF EXISTS (SELECT * FROM table2) SELECT @x = -1 END GO</pre>

Oracle	Sybase ASE
<p>Example 4:</p> <pre> CREATE OR REPLACE PROCEDURE procl (v_basesal IN NUMBER(19,2), v_empid IN NUMBER) AS v_temp NUMBER(1, 0) := 0; BEGIN SELECT 1 INTO v_temp FROM DUAL WHERE (SELECT sal FROM emp WHERE empid = v_empid) < v_basesal; IF v_temp = 1 THEN UPDATE emp SET sal_flag = -1 WHERE empid = v_empid; END IF; END;</pre>	<p>Example 4:</p> <pre> DROP PROC procl GO CREATE PROC procl @basesal money, @empid int AS BEGIN IF (select sal from emp where empid = @empid) < @basesal UPDATE emp SET sal_flag = -1 WHERE empid = @empid END GO</pre>

9.6.5.4.1 COMMENTS:



IF statements in Oracle and Sybase ASE are nearly the same except in the following two cases:

- Sybase ASE allows you to probe query results with the EXISTS function as well as singled column select results that return exactly one row with one column result. In this case you can put the entire SELECT statement into (...) and compare it to a constant or variable.

9.6.5.5 RETURN STATEMENT

Oracle	Sybase ASE
<p>Example 1:</p> <pre>CREATE OR REPLACE PROCEDURE proc1 (v_x IN NUMBER v_s OUT NUMBER) AS BEGIN IF v_x = -1 THEN v_s := 25022; ELSE v_s := 25011; END IF; END;</pre>	<p>Example 1:</p> <pre>DROP PROC proc1 GO CREATE PROC proc1 @x int AS IF @x = -1 RETURN 25022 ELSE RETURN 25011 GO</pre>
<p>Example 2:</p> <pre>CREATE OR REPLACE FUNCTION func1 (v_x IN NUMBER) RETURN VARCHAR2 AS BEGIN IF v_x = -1 THEN RETURN 'This is an error'; ELSE RETURN 'This is great'; END IF; END;</pre>	<p>Example 2:</p> <pre>DROP FUNC func1 GO CREATE FUNCTION func1 (@x int) RETURNS varchar(20) AS BEGIN DECLARE @bonus varchar(20) SET @bonus = 0 IF (@x < 10) SET @bonus = 'This is an error' ELSE SET @bonus = 'This is great' END RETURN @bonus END GO</pre>

9.6.5.5.1 COMMENTS:

A RETURN statement is used to return a single value back to the calling program and works the same in both databases. Sybase ASE can return only the numeric data type, while Oracle can return any of the server data types or the PL/SQL data types.



If an Oracle function returns a numeric value, this can be converted into a Sybase ASE procedure. If it is any other data type a Sybase ASE function is needed.

9.6.5.6 RAISERROR STATEMENT

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE proc1 AS BEGIN raise_application_error(-20999, 'No Employees found'); END;</pre>	<pre>DROP PROC proc1 GO CREATE PROC proc1 AS RAISERROR 12345 "No Employees found" GO</pre>

9.6.5.6.1 COMMENTS:

Oracle supports a multitude of exception handlings and error mechanism. Sybase ASE only supports the RAISERROR statement, because Oracle checks each SQL statement for errors before proceeding to the next and Sybase ASE passes the control to the next SQL statement, even in the case of an error.

The closes Oracle error handling to the Sybase ASE RAISERROR statement is the `raise_application_error` function. All the other Oracle error handling functions need to either be recreated in Sybase ASE with the same function name or dropped from the code.



Recreating the Oracle predefined PL/SQL exception into Sybase ASE procedures or function, while adding the RAISERROR statement at the end of this procedure or function to stop processing, is probably the best approach.

There are also Oracle user-defined PL/SQL exceptions that need to be converted the same way.

9.6.5.7 EXECUTE STATEMENT

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE proc1 AS BEGIN SetExistFlag; SetExistFlag(yes=>v_yes,Status); Status:=RecordExists; SetExistFlag(v_yes); END;</pre>	<pre>DROP PROC proc1 GO CREATE PROC proc1 AS BEGIN EXEC SetExistFlag EXEC SetExistFlag yes=@yes, @Status OUT EXEC @Status = RecordExists EXEC SetExistFlag @yes END GO</pre>

9.6.5.7.1 COMMENTS:



Both Oracle and Sybase ASE can execute stored procedures by calling its name. Sybase ASE has the restriction that if you call more than one stored procedure from the same SQL statement block, the stored procedure name must be preceded by the word EXEC.

9.6.5.8 WHILE STATEMENT

Oracle	Sybase ASE
<p>Example1:</p> <pre> CREATE OR REPLACE PROCEDURE procl (iv_i IN NUMBER) AS v_i NUMBER(10,0) :=iv_i; BEGIN WHILE v_i > 0 LOOP BEGIN DBMS_OUTPUT.PUT_LINE('Looping inside WHILE....'); v_i := v_i + 1; END; END LOOP; END;</pre>	<p>Example1:</p> <pre> DROP PROC procl GO CREATE PROC procl @i int AS BEGIN WHILE @i > 0 BEGIN PRINT 'Looping inside WHILE....' SELECT @i = @i + 1 END END GO</pre>
<p>Example2:</p> <pre> CREATE OR REPLACE PROCEDURE procl (iv_i IN NUMBER, v_y IN NUMBER) AS v_i NUMBER(10,0):=iv_i; BEGIN WHILE v_i > 0 LOOP BEGIN DBMS_OUTPUT.PUT_LINE('Looping inside WHILE....'); v_i := v_i + 1; END; END LOOP; END;</pre>	<p>Example2:</p> <pre> DROP PROC procl GO CREATE PROC procl @i int, @y int AS BEGIN WHILE @i > 0 BEGIN PRINT 'Looping inside WHILE....' SELECT @i = @i + 1 END END GO</pre>
<p>Example3:</p> <pre> CREATE OR REPLACE PROCEDURE procl AS v_sal NUMBER(19,2); v_temp NUMBER(1, 0) := 0; BEGIN v_sal := 0; LOOP v_temp := 0; SELECT 1 INTO v_temp FROM DUAL</pre>	<p>Example3:</p> <pre> DROP PROC procl GO CREATE PROC procl AS BEGIN DECLARE @sal money SELECT @sal = 0 WHILE EXISTS(SELECT * FROM emp where sal < @sal) BEGIN</pre>

Oracle

```
WHERE EXISTS ( SELECT *
                FROM emp
                WHERE sal < v_sal );
IF v_temp != 1 THEN
    EXIT;
END IF;
BEGIN
    v_sal := v_sal + 99;
    DELETE emp
    WHERE sal < v_sal;
END;
END LOOP;
END;
```

Sybase ASE

```
SELECT @sal = @sal + 99
DELETE emp
WHERE sal < @sal
END
END
GO
```

Example4:

```
CREATE OR REPLACE PROCEDURE procl
AS
    v_sal NUMBER(19,2);
    v_temp NUMBER(1, 0) := 0;
BEGIN
    LOOP
        v_temp := 0;
        SELECT 1 INTO v_temp
        FROM DUAL
        WHERE ( SELECT COUNT(*)
                FROM emp ) > 0;
        IF v_temp != 1 THEN
            EXIT;
        END IF;
        BEGIN
            SELECT MAX(sal)
            INTO v_sal
            FROM emp
            WHERE stat = 1;
            DELETE emp
            WHERE sal < v_sal;
        END;
    END LOOP;
END;
```

Example4:

```
DROP PROC procl
GO

CREATE PROC procl
AS
BEGIN
    DECLARE @sal money
    WHILE (SELECT count (*)
           FROM emp ) > 0
    BEGIN
        SELECT @sal = max(sal)
        FROM emp
        WHERE stat = 1
        DELETE emp
        WHERE sal < @sal
    END
END
GO
```

9.6.5.8.1 COMMENTS:



The WHILE statement in both Oracle and Sybase ASE is very similar. The same exceptions and rules apply as in the IF statements.

9.6.5.9 GOTO STATEMENT

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE procl (v_Status IN NUMBER) AS v_j NUMBER(10,0); BEGIN IF v_Status = -1 THEN GOTO Error; END IF; v_j := -1; <<Error>> v_j := -99; END;</pre>	<pre>DROP PROC procl GO CREATE PROC procl @Status int AS BEGIN DECLARE @j int IF @Status = -1 GOTO Error SELECT @j = -1 Error: SELECT @j = -99 END GO</pre>

9.6.5.10 SQL%ROWCOUNT AND SQLCODE VS. @@ROWCOUNT AND @@ERROR VARIABLES

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE procl AS v_sys_error NUMBER := 0; v_x NUMBER(10,0); BEGIN BEGIN SELECT COUNT(*) INTO v_x FROM emp ; EXCEPTION WHEN OTHERS THEN v_sys_error := SQLCODE; END; IF SQL%ROWCOUNT = 0 THEN DBMS_OUTPUT.PUT_LINE('No rows found.');</pre>	<pre>DROP PROC procl GO CREATE PROC procl AS BEGIN DECLARE @x int SELECT @x=count(*) FROM emp IF @@rowcount = 0 print 'No rows found.' IF @@error = 0 print 'No errors.'</pre>
<pre>END IF; IF v_sys_error = 0 THEN DBMS_OUTPUT.PUT_LINE('No errors.');</pre>	<pre>END GO</pre>
<pre>END IF; END;</pre>	

9.6.5.10.1 COMMENTS:



The Oracle SQL%ROWCOUNT can be translated into Sybase ASE @@rowcount and the SQLCODE in PL/SQL would be the equivalent of @@error in Sybase ASE.

9.6.5.11 ASSIGNMENT STATEMENT

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE proc1 AS v_x NUMBER(10,0); BEGIN v_x := -1; SELECT SUM(salary) INTO v_x FROM emp ; END;</pre>	<pre>DROP PROC sp_proc1 GO CREATE PROC sp_proc1 AS BEGIN DECLARE @x int SELECT @x = -1 SELECT @x=sum(salary) FROM Emp END GO</pre>

9.6.5.11.1 COMMENTS:

PL/SQL assigns values to a variable by using the assignment statement to assign the value of a variable or an expression to a local variable. It assigns a value from a database using the SELECT..INTO clause. This requires that the SQL returns only one row.



An assignment of variables in Sybase ASE is done via a SELECT statement.

9.6.5.12 SELECT STATEMENT

Retrieving data from a stored procedure in a Sybase ASE is very different from Oracle. The two examples below illustrate how a single value is being returned vs. handling of an entire result set. The biggest difference is that Sybase ASE does not support the REFCURSOR data type to return result sets.



The Sybase ASE execute command does not support multiple results sets. This means that stored procedures call from within another stored procedure or an isql command line cannot return multiple result sets. Stored procedures called by JDBC or ODBC can return multiple result sets.

The main reason for using stored procedures to embed SELECT statements is security. To enable the same security in Sybase ASE you need to create this extra layer through a view.

Oracle	Sybase ASE
<p>Example 1:</p> <pre> CREATE OR REPLACE PROCEDURE getOrder(oNum IN VARCHAR2, OHeader IN OUT SYS_REFCURSOR) IS BEGIN OPEN oHeader FOR SELECT * FROM ORDER WHERE ORDER_NO = oNum; END getOrder; / CREATE OR REPLACE package order_pkg AS PROCEDURE getOrderDetails(oNum IN VARCHAR2, ODetail IN OUT SYS_REFCURSOR); END order_pkg; / CREATE OR REPLACE PACKAGE BODY order_pkg AS PROCEDURE getOrderDetails(oNum IN VARCHAR2, ODetail IN OUT SYS_REFCURSOR) IS BEGIN OPEN oDetail FOR SELECT * FROM ORDER_LINEITEM WHERE L_ORDER_NO = oNum; END getOrderDetails; END order_pkg; / -- Anonymous PL/SQL block set serveroutput on; BEGIN DECLARE h SYS_REFCURSOR; </pre>	<p>Example 1:</p> <pre> CREATE VIEW oHeader AS SELECT * FROM ORDER GO CREATE VIEW oDetail AS SELECT * FROM ORDER_LINEITEM GO BEGIN DECLARE @order_no varchar(32), @BILLTO_NAME varchar(50), @CC_NO varchar(50), @BILLTO_DT datetime, @d_order_no varchar(32), @d_LINE_NUM int, @d_PRODUCT varchar(40), @d_QUANTITY int, @d_UNIT_AMT money, DECLARE h CURSOR FOR SELECT * FROM oHeader OPEN h FETCH h into @order_no, @BILLTO_NAME, @CC_NO, @BILLTO_DT WHILE (@@SQLTATUS = 0) BEGIN PRINT convert(char(35),@order_no)+' '+convert(char(55),@BILLTO_NAME)+' '+convert(char(55),@CC_NO)+' '+convert(varchar(35),@BILLTO_DT,121) DECLARE d CURSOR FOR SELECT * FROM oDetail </pre>

```

d          SYS_REFCURSOR;
order_no   VARCHAR2(32);
BILLTO_NAME VARCHAR2(50);
CC_NO      VARCHAR2(50);
BILLTO_DT   DATE;
d_order_no  VARCHAR2(32);
d_line_num  NUMBER;
d_product   VARCHAR2(40);
d_quantity  NUMBER;
d_unit_amt  NUMBER;

BEGIN
  getOrder('999',h);
  LOOP
    FETCH h INTO order_no, BILLTO_NAME,
    CC_NO, BILLTO_DT;
    EXIT WHEN h%NOTFOUND;

    dbms_output.put_line(RPAD(order_no,35)||RPAD
    (billto_name,55)|| LPAD(cc_no,55)||' ' ||
    TO_CHAR(billto_dt,'YYYYMMDD'));
    order_pkg.getOrderDetails(order_no,d);
    LOOP
      FETCH d INTO
    d_order_no,d_line_num,d_product,d_quantity,d
    _unit_amt;
      EXIT WHEN d%NOTFOUND;
      dbms_output.put_line(' ' ||
    RPAD(d_order_no,35)||LPAD(TO_CHAR(d_line_num
    , '99'),5) ||' ' || RPAD(d_product,43)||
    TO_CHAR(d_quantity,'999,999') ||' ' ||
    TO_CHAR(d_unit_amt,'999,990.00'));
    END LOOP;
    CLOSE d;
  END LOOP;
  CLOSE h;
END;
END;
/

```

```

OPEN d
  FETCH d into @d_order_No, @d_line_num,
@d_product, @d_quantity, @d_unit_amt
  WHILE (@@SQLTATUS = 0)
  BEGIN
    PRINT convert(char(35),@order_no)+'
'+convert(char(2),@d_line_num)+' '+@d_product+'
'+convert(char(15),@d_quantity)+'
'+convert(char(15),@d_unit_amt)
    FETCH d into @d_order_No, @d_line_num,
@d_product, @d_quantity, @d_unit_amt
  END
  CLOSE d
  DEALLOCATE CURSOR d
  FETCH h into @order_no, @BILLTO_NAME, @CC_NO,
@BILLTO_DT
  END
  CLOSE h
  DEALLOCATE CURSOR h
END
GO

```

Example2:

```

CREATE OR REPLACE FUNCTION func1
AS
  v_name CHAR(20);
BEGIN
  SELECT id
  INTO v_id
  FROM emp;
  RETURN v_id;
END;

```

Example2:

```

DROP PROC proc1
GO

CREATE PROC proc1
AS
BEGIN
  DECLARE @name char(20)
  SELECT @id = id
  FROM emp
  RETURN id
END
GO

```


9.6.5.12.1 COMMENTS:



Because of the differences in their architectures, Sybase ASE stored procedures return data to the client program in a different way than Oracle. Oracle and Sybase ASE can all pass data to the client using output parameters in the stored procedures.

9.6.5.13 COLUMN ALIAS

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE procl (v_Status IN NUMBER DEFAULT 0, cv_1 IN OUT SYS_REFCURSOR) AS BEGIN OPEN cv_1 FOR SELECT SUM(salary) x FROM emp; END;</pre>	<pre>DROP PROC sp_procl GO CREATE PROC sp_procl @Status int=0 AS BEGIN SELECT sum(salary) as x FROM emp END GO</pre>

9.6.5.14 TEMPORARY TABLES

Oracle	Sybase ASE
<pre>CREATE OR REPLACE PROCEDURE proc1 AS BEGIN DELETE FROM tt_Tab; INSERT INTO tt_Tab (SELECT col1, col2 FROM table1 WHERE table1.id = 100); END;</pre>	<pre>DROP PROC sp_proc1 GO CREATE PROC sp_proc1 AS BEGIN SELECT col1, col2 INTO #Tab FROM table1 WHERE table1.id = 100 END GO</pre>

9.6.5.14.1 COMMENTS:



The main difference between Oracle and Sybase ASE is on how the systems handle query results. Sybase ASE is based on so called 'set processing', meaning that in a stored procedure result sets are typically stored in temporary tables and then further refined, whereas Oracle is based on cursor processing navigating through result sets. The underlying reason is that Oracle maintains a versioning of its transactions and guarantees data integrity through cursors, Sybase ASE on the other hand does not support versioning, but you can think of temporary tables as a kind of versioning.

You will hardly find many examples of Oracle temporary tables and therefore conversion issues are very low.

9.6.5.15 CURSOR HANDLING

Oracle	Sybase ASE
<pre> CREATE OR REPLACE PROCEDURE cursor_demo AS CURSOR cursor_1 IS SELECT empno, ename, sal FROM emp ; v_empno NUMBER(10,0); v_ename CHAR(100); v_sal NUMBER; BEGIN OPEN cursor_1; FETCH cursor_1 INTO v_empno,v_ename,v_sal; CLOSE cursor_1; END; </pre>	<pre> DROP PROC sp_procl GO CREATE PROC sp_procl AS BEGIN DECLARE @empno INT DECLARE @ename CHAR(100) DECLARE @sal FLOAT DECLARE cursor_1 CURSOR FOR SELECT empno, ename, sal FROM emp OPEN cursor_1 FETCH cursor_1 INTO @empno, @ename, @sal CLOSE cursor_1 DEALLOCATE CURSOR cursor_1 END GO </pre>

9.6.5.15.1 COMMENTS:



Both Oracle and Sybase ASE support cursors. But there are some differences. Oracle PL/SQL is implemented with an implicit cursor deallocation process. When you close an Oracle cursor, it gets automatically deallocated. Sybase ASE requires an explicit deallocate cursor statement to do so. Both Oracle and Sybase ASE support scrollable cursor.

Sybase ASE supports 2 levels of cursor sensitivity:

- **insensitive:** An insensitive cursor is a snapshot of the result set, taken when the cursor is opened. An internal worktable is created and fully populated with the cursor result set when you open the cursor. Any locks on the base tables are released, and only the worktable is accessed when you execute fetch. Any data changes in the base table on which the cursor is declared do not affect the cursor result set.

This is the Sybase ASE cursor sensitivity that is closest to Oracle's cursor implementation.

- **semi_sensitive:** In a semi_sensitive cursor, some data changes in the base tables may appear in the cursor. The query plan chosen and whether the data rows have been fetched at least

once may affect the visibility of the base table data change.

semi_sensitive scrollable cursors are like insensitive cursors, in that they use a worktable to hold the result set for scrolling purposes. In semi_sensitive mode, the cursor's worktable materializes as the rows are fetched, rather than when you open the cursor. The membership of the result set is fixed only after all the rows have been fetched once, and copied to the scrolling worktable.

This is the **default** cursor sensitivity in Sybase ASE.

9.6.6 PL/SQL VS. T/SQL LANGUAGE ELEMENTS

9.6.6.1 TRANSACTION HANDLING SEMANTICS

Oracle applies ANSI-standard implicit transaction methods. A logical transaction begins with the first executable SQL statement after a COMMIT, ROLLBACK, or connection to the database. A transaction ends with a COMMIT, ROLLBACK, or disconnection from the database. An implicit COMMIT statement is issued before and after a DDL statement. The implicit transaction model prevents artificial nesting of transactions because only one logical transaction per session can be in effect. The user can set SAVEPOINT in a transaction and roll back a partial transaction to the SAVEPOINT.

For example:

```
UPDATE test_table SET coll = 'value1';
SAVEPOINT first_sp;
UPDATE emp SET coll = 'value2';
ROLLBACK TO SAVEPOINT first_sp;
COMMIT; /* coll is 'value1' */
```



Sybase ASE offers two different transaction models; the ANSI-standard implicit transaction model and the explicit transaction model. Sybase ASE provides options to support ANSI-standard transactions. These options can be set or un-set using the SET command.

The following SET command sets the implicit transaction mode:

```
set chained on
```

This will eliminate the need to add `BEGIN TRAN` statements at the beginning of a session and after every `COMMIT` or `ROLLBACK` statement. But it also creates long and impact the ability to add new code and using Sybase ASE terminology, like nesting transactions. Unlike Oracle, Sybase ASE supports nested transactions and this is the preferred method to provide savepoints and roll back a partial transaction.

You cannot execute the `set chained` command within a transaction. To return to the unchained transaction mode, set the `chained` option to off. The default transaction mode is unchained.

In chained transaction mode, Adaptive Server implicitly executes a `begin transaction` statement just before the following data retrieval or modification statements: `delete`, `insert`, `open`, `fetch`, `select`, and `update`. For example, the following group of statements produces different results, depending on which mode you use:

```
insert into publishers
  values ("9906", null, null, null)
begin transaction
delete from publishers where pub_id = "9906"
rollback transaction
```

In unchained transaction mode, the `rollback` affects only the `delete` statement, so *publishers* still contains the inserted row. In chained mode, the `insert` statement implicitly begins a transaction, and the `rollback` affects all statements up to the beginning of that transaction, including the `insert`.

The following command sets the isolation level to the desired level:

```
set transaction isolation level (1|3)
```

Isolation level 1 prevents dirty reads. Isolation level 2 prevents un-repeatable reads. Isolation level 3 prevents phantoms. Isolation level 3 is required by ANSI standards. The default for Sybase ASE is isolation level 1.



To implement isolation level 3, Sybase ASE applies `HOLDLOCK` to all tables taking part in the transaction. Be aware that Sybase ASE does apply shared read locks and if a `HOLDLOCK` is in progress, these locks cannot be applied, therefore the application that tries to read will be blocked for the duration of the transaction the issued the `HOLDLOCK`. This will cause serious performance issues.

Using the `set chained on` option in combination with the `set isolation level 3` option provides the environment in Sybase ASE that is closely compatible to Oracle's transaction method base on the

ANSI-standard. This might be the easy way to make an Oracle transaction work in Sybase ASE without any changes. But the trade-off is huge, with blocking transactions and performance degradations.

9.6.6.1.1 COMPARISON OF TRANSACTION-HANDLING STATEMENTS IN ORACLE AND SYBASE ASE

Oracle	Sybase ASE
	BEGIN TRAN
SAVEPOINT tran_1	BEGIN TRAN tran_1
COMMIT	COMMIT TRAN
COMMIT	COMMIT TRAN tran_1
ROLLBACK	ROLLBACK TRAN
ROLLBACK TO SAVEPOINT tran_1	ROLLBACK TRAN tran_1

9.6.6.2 EXCEPTION-HANDLING AND ERROR-HANDLING SEMANTICS

In Oracle, each SQL statement is automatically checked for errors before proceeding with the next statement. If an error occurs, control immediately jumps to an exception handler if one exists. For example, if a SELECT statement does not find any row in the database, an exception is raised. The corresponding exception handler part of the block includes the code to deal with this error. The built-in RAISE_APPLICATION_ERROR procedure notifies the client of the server error condition and returns immediately to the calling routine.

Oracle places an implicit SAVEPOINT at the beginning of a procedure. The built-in RAISE_APPLICATION_ERROR procedure rolls back to this SAVEPOINT or the last committed transaction within the procedure. The control is returned to the calling routine.



Sybase ASE passes the control from one SQL statement to another without checking for errors. This means that you have to check for errors after every SQL statement within a procedure. Sybase ASE's equivalent to Oracle's RAISE_APPLICATION_ERROR is called RAISERROR. Unlike Oracle, RAISERROR does not return the controls to the calling routine.

The first step in the conversion is to replace all RAISE_APPLICATION_ERROR calls with RAISERROR calls, followed immediately with a RETURN statement to emulate the Oracle exception handling.

The second step is to deal with the implicit SAVEPOINTS Oracle creates at the beginning of each procedure. If the transaction is within one procedure, no problem, but if the procedures are nested you will need to apply nested transactions to emulate a similar behavior. Applying nested transaction implies that the transactions are no longer ANSI-standard and that implicit BEGIN TRAN statements have to be used instead of a single `set chained on` option.

9.6.6.3 SPECIAL GLOBAL VARIABLES

Oracle supports many special global variables with PL/SQL as well as Sybase ASE does for T-SQL. For the purpose of converting PL/SQL procedures to T-SQL procedures, only two are of interest.

Oracle	Sybase ASE
SQLCODE	@@error
The server error code indicating the execution status of	The server error code indicating the execution status of

the most recently executed PL/SQL statement.	the most recently executed T-SQL statement.
SQL%ROWCOUNT	@@rowcount
The number of rows affected by the most recently executed PL/SQL statement.	The number of rows affected by the most recently executed T-SQL statement.

9.6.6.4 DDL CONSTRUCT

Oracle allows DDL statement as part of the dynamic SQL. Oracle issues an implicit COMMIT statement after each DDL statement. Sybase ASE allows DDL statements as part of a transaction only if the 'ddl in tran' database option is turned on. By default this option is turned off. To enable DDL in transaction the following command has to be executed:

```
sp_dboption database_name,"ddl in tran", true
```

Then execute the `checkpoint` command in that database.



To achieve the same transaction behavior in Sybase ASE as in Oracle, a COMMIT statement has to be added after each DDL statement. If the non ANIS-standard transaction is in use, a BEGIN TRAN has to follow immediately after the COMMIT.

10 VALIDATE 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 Sybase ASE installation, security, database objects, data, and performance.

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

10.1 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.

10.2 SECURITY

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

10.3 LOGICAL ARCHITECTURE OF THE DATABASE

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

10.4 DATA

Validate the number and type of objects in Oracle and Sybase ASE 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.

10.5 FUNCTIONALITY

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

10.6 PERFORMANCE

The response time and throughput of the application using Sybase ASE should be same or better than the application with Oracle database.

11 SYBASE ASE FEATURES

There are several ASE features and tools that may not be available in Oracle that the DBA/developer should consider using during development and production of the application.

11.1 ADDITIONAL ASE FEATURES

11.1.1 LOCKING SCHEME

The Sybase ASE server gives the developer the possibility to choose the lock scheme of the table. The options are:

1. **Allpages Locking** – Allpages locking locks both data pages and index pages. When a query updates a value in a row in an allpages-locked table, the data page is locked with an exclusive lock. Any index pages affected by the update are also locked with exclusive locks. These locks are transactional, meaning that they are held until the end of the transaction.
2. **Datapages Locking** – In datapages locking, entire data pages are still locked, but index pages are not locked. When a row needs to be changed on a data page, that page is locked, and the lock is held until the end of the transaction. The updates to the index pages are performed using latches, which are non-transactional. Latches are held only as long as required to perform the physical changes to the page and are then released immediately. Index page entries are implicitly locked by locking the data page. No transactional locks are held on index pages.
3. **Datarows Locking** – In datarows locking, row-level locks are acquired on individual rows on data pages. Index rows and pages are not locked. When a row needs to be changed on a data page, a non-transactional latch is acquired on the page. The latch is held while the physical change is made to the data page, and then the latch is released.

The lock on the data row is held until the end of the transaction. The index rows are updated, using latches on the index page, but are not locked. Index entries are implicitly locked by acquiring a lock on the data row.

Allpage is the best choice for tables that are read-only or tables that are not frequently accessed by a lot of users.



Datarow should be chosen for tables that frequently accessed/changed by a lot of users.

11.1.2 MEMORY MANAGEMENT

Sybase ASE offers sophisticated memory management. When you install Sybase ASE, it has a single default data cache that is used for all data, index, and log activity. The data cache for the server can be split up into a number of distinct “*Named Caches*”. Creating other caches does not reduce the size of the default data cache and does not divide the default cache into separate cache structures. You can then bind a database, table (including the *syslogs* table), index, or text or image page chain to a named data cache. The named data caches that you create can be used only by databases or database objects that are explicitly bound to them. All objects not explicitly bound to named data caches use the default data cache. With the named data cache, different aspects of the server’s workload can be separated so that they do not interfere with each other. For example, using named caches can prevent OLTP workload flushing DSS workload’s pages from the data cache. Another use of named caches is to keep frequently used tables in memory by creating a cache for each table and only binding the one table to the cache (thus preventing the table’s pages ever being flushed from memory).

In addition, a cache may be further subdivided into “*Pools*” which each perform different sized disk I/O. Pools may be set up to provide 2K, 4K, 8K or 16K I/Os. The server will automatically select the appropriate disk I/O size from the available pools in the data cache it is currently using, depending on the nature of the query it is processing. The commands that move space between pools within a cache do not require a restart of Sybase ASE, so you can reconfigure pools to meet changing application loads with little impact on server activity. By using named caches and pools to control I/O size, an Sybase ASE Enterprise DBA can exercise great control over the management of memory and I/O for their server, thus supporting their unique workloads effectively.

Sybase ASE allocates physical memory dynamically. This allows users to change the memory configuration of Sybase ASE without restarting the server.

11.1.2.1 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 requires 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. Sybase ASE dynamically allocates and frees memory during the task’s runtime. The heap memory pool dramatically reduces the pre-declared 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.

Adaptive Enterprise server performance can be improved dynamically changing the configuration parameters related to memory.

- Adding a new cache
- Adding memory to the existing cache
- Deleting a cache
- Changing a cache type

11.1.3 LOGICAL PROCESS MANAGER

The Logical Process Manager introduces the ability to define a number of user-defined priority task queues, each for a different type of work. It allows you to assign priorities to logins, procedures, or applications using **sp_bindexclass** and related system procedures.

Each task has a priority assigned to it; the priority can change over the life of the task. When an engine looks for a task to run, it first scans its own high-priority queue and then the high-priority global run queue.

If there are no high-priority tasks, it looks for tasks at medium priority, then at low priority. If it finds no tasks to run on its own run queues or the global run queues, it can examine the run queues for another engine, and steal a task from another engine. This combination of priorities, local and global queues, and the ability to move tasks between engines when workload is uneven provides load balancing.

Tasks in the global or engine run queues are all in a runnable state. Output from **sp_who** lists tasks as "runnable" when the task is in any run queue.

11.1.4 USER DEFINED ROLES

In addition to the system-defined roles (such as `sa_role` and `sso_role`), ASE provides the facility for "User Defined Roles." These roles are created by the DBA and can be granted database permissions (in the same way that permissions can be granted to users and groups). The roles can also be granted to

each other, thus providing the ability to create a permissions hierarchy (where the role pediatricians inherits all of the permissions from the doctors role, for example). Use of user-defined roles allows a DBA to create a powerful and flexible security model, tailored to their application's requirements.

11.1.5 DBCC

The database consistency checker – **dbcc** provides commands for checking the logical and physical consistency of a database. Major functions of **dbcc** are:

- Checking page linkage and data pointers at both the page level and the row level using **checkstorage** or **checktable** and **checkdb**
- Checking page allocation using **checkstorage**, **checkalloc**, **checkverify**, **tablealloc**, and **indexalloc**
- Checking for consistency within and between the system tables in a database with **checkcatalog** **dbcc** checkstorage stores the results of checks in the *dbccdb* database. By using **dbcc** stored procedure, reports from *dbccdb* can be printed **dbcc** commands can be used:
- As part of regular database maintenance—the integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency checks on a regular basis.
- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database for additional confidence in the integrity of the backup.
- If you suspect that a database is damaged. For example, if using a particular table generates the message "Table corrupt," you can use **dbcc** to determine if other tables in the database are also damaged.

If you are using Component Integration Services, there are additional **dbcc** commands you can use for remote databases.

The **dbcc** commands for checking the consistency of databases and tables are:

- **dbcc checkstorage**
- **dbcc checktable**
- **dbcc checkdb**

The **dbcc checkstorage** to perform the following checks:

- Allocation of text valued columns
- Page allocation and consistency
- OAM (Object Allocation Map) page entries
- An OAM page for each partition exists
- Pointer consistency
- Text-valued columns and text-column chains

The **dbcc checktable** checks the specified table to see that:

- Index and data pages are linked correctly
- Indexes are sorted properly
- Pointers are consistent
- All indexes and data partitions are correctly linked
- Data rows on each page have entries in the row-offset table; these entries match the locations for the data rows on the page
- Partition statistics for partitioned tables are correct

11.1.6 PARALLEL QUERY

Sybase ASE supports horizontal, vertical and pipelined parallelism for query execution.

Vertical parallelism is the ability to run multiple operators at the same time by employing different system resources such as CPUs, disks, and so on.

Horizontal parallelism is the ability to run multiple instances of an operator on the specified portion of the data.

Pipelining is a form of vertical parallelism in which intermediate results are piped to higher operators in a query tree. The output of one operator is used as input for another operator. The operator used as input can run at the same time as the operator feeding the data, which is an essential element in pipelined parallelism.

When Sybase ASE is configured for parallel query processing, the query optimizer evaluates each query to determine whether it is eligible for parallel execution. If it is eligible, and if the optimizer determines that a parallel query plan can deliver results faster than a serial plan, the query is divided into plan fragments that are processed simultaneously. The results are combined and delivered to the client in a shorter period of time than it would take to process the query serially as a single fragment. Parallel query processing can improve the performance of these types of queries:

- **select** statements that scan large numbers of pages but return relatively few rows, such as table scans or clustered index scans with grouped or ungrouped aggregates.
- Table scans or clustered index scans that scan a large number of pages, but have **where** clauses that return only a small percentage of rows.
- **select** statements that include **union**, **order by**, or **distinct**, since these query operations can make use of parallel sorting or parallel hashing.
- **select** statements where a reformatting strategy is chosen by the optimizer, since these can populate worktables in parallel and can make use of parallel sorting.
- **join** queries also benefit from parallel access.

The degree of parallelism can be set at session level or in stored procedures or triggers or in a query.

11.1.7 BACKUP SERVER

This separate server works in conjunction with the ASE server to perform backup and recovery processing on behalf of the ASE server. This arrangement has the advantage of moving the effort required to perform backup and restore operations from the query processing server to a specialist server built and tuned for that purpose. The Backup Server is accessed automatically by the ASE server as part of its processing of the dump and load commands. The DBA does not need to refer to the Backup Server explicitly. As well as simply being more efficient at performing backup and recovery operations than the ASE server, the Backup Server offers a number of other advantages. In particular, it provides a way to back up and restore many devices simultaneously (a process known as “striping”) and it allows these devices to be attached to any machine on the network. Backup Server can dump to a maximum of 32 stripes, either disk or tape. The backup device can be any writeable device including tape, disk files, optical media etc. (Backup Server automatically senses the device’s characteristics, as required,) In addition, a Backup Server API is available to partner companies to allow them to integrate more complex devices (such as tape stackers) smoothly into the Sybase environment. In summary, Backup Server brings an unprecedented degree of flexibility and power to database server backup and restore operations.

11.1.8 ASE WITH ENCRYPTED COLUMNS

Sybase ASE authentication and access control mechanisms ensure that only properly identified and authorized users can access data. Data encryption further protects sensitive data on disk or in archives against theft and security breaches.

Data encryption in Sybase ASE allows you to encrypt data at the column level. This allows you to encrypt only sensitive data, which minimizes processing overhead.

Encrypting columns in Sybase ASE is more straightforward than using encryption in the middle tier, or in the client application. You use SQL statements to create the encryption keys and specify columns for encryption. Sybase ASE handles key generation and storage. Encryption and decryption of data occurs automatically and transparently as you write and read the data in encrypted columns. No application changes are required, and there is no need to purchase third-party software.

Sybase ASE generates encryption keys and stores them in the database in encrypted form. Key owners grant table owners permission to encrypt columns with a named key.

All the information related to keys and encryption is encapsulated by the **create encryption key**, which allows you to specify the key's name, the encryption algorithm, the key size, the key's **default** property, as well as whether an initialization vector or padding is used during the encryption process. See below for the **create encryption key** syntax.

Column encryption in Sybase ASE uses the Advanced Encryption Standard (AES) symmetric key encryption algorithm, with available key sizes of 128, 192, and 256 bits. The Security Builder Crypto API provides random-key generation and cryptographic functionality.

You can create separate keys for each encrypted column. Keys can be shared between columns, but each column can have only one key.

The System Security Officer can set up a default encryption key for the database. The default key is used whenever the **encrypt** qualifier is used without a key name on **create table**, **alter table**, and **select into**.

To securely protect key values, Sybase ASE uses the system encryption password to generate a 128-bit key-encrypting key, which in turn is used to encrypt the newly created key (this is the column encryption key.) The column-encryption key is stored in encrypted form in the *sysencryptkeys* system table.

When you specify a column for encryption, you can use either a named key from the same database, or from a different database. If you do not specify a named key, the column is automatically encrypted with the default key from the same database.

Encrypting with a key from a different database provides a security advantage because, in the event of the theft of a database dump, it protects against access to both keys and encrypted data.

Administrators can also protect each database dump with a different password, making unauthorized access even more difficult.

Encrypting with a key from a different database needs special care to avoid data and key integrity problems in distributed systems. Carefully coordinate database dumps and loads. If you use a named key from a different database, Sybase recommends that, when you:

- Dump the database containing encrypted columns; you also dump the database where the key was created. You must do this if new keys have been added since the last dump.
- Dump the database containing an encryption key; dump all databases containing columns encrypted with that key. This keeps encrypted data in sync with the available keys.

11.1.9 PASSWORD PROTECTED BACKUPS

Database dump can be protected from unauthorized loads using the password parameter of the dump database command. If you include the password parameter when you make a database dump, you must also include this password when you load the database.

11.1.10 ENHANCED FULL-TEXT SEARCH SPECIALTY DATA STORE

The Enhanced Full-Text Search Specialty Data Store product performs powerful, full-text searches on Sybase ASE data. In Sybase ASE, without the Enhanced Full-Text Search engine, you can search text columns only for data that matches what you specify in a select statement. For example, if a table contains documents about dog breeds, and you perform a search on the words "Saint Bernard," the query produces only the rows that include "Saint Bernard" in the text column.

Enhanced Full-Text Search Specialty Data Store is an Open Server™ application built on the Verity technology that is available in the Verity Developer's Kit. Sybase ASE connects to the Enhanced Full-Text Search engine through Component Integration Services (CIS), allowing queries written in the Verity query language to perform full-text searches on Sybase ASE data.

With the Enhanced Full-Text Search engine, you can expand queries on text columns to:

- Rank the results by order of how often a searched item appears in the selected document. For example, you can obtain a list of document titles that reference the words "Saint Bernard" five or more times.
- Select documents in which the words you search for appear within n number of words of each other. For example, you can search only for the documents that include the words "Saint Bernard" and "Swiss Alps" and that appear within 10 words of each other.

- Select documents that include all the search elements you specify within a single paragraph or sentence. For example, you can query the documents that include the words "Saint Bernard" in the same paragraph or sentence as the words "Swiss Alps."
- Select documents that contain one or more synonyms of the word you specify. For example, you can select documents that discuss "dogs," and it returns documents that contain the words "dogs," "canine," "pooch," "pup," and so on.
- Create your own custom thesaurus. For example, you can create a custom thesaurus that includes "working dogs," "St. Bernard," "large dogs," and "European Breeds" as synonyms for "Saint Bernard."
- Create topics that specify the search criteria for a query. For example, you can create a topic that returns documents that include the phrase "Saint Bernard" or "St. Bernard," followed by documents that include the phrase "working dogs," "large dogs," or "European Breeds."
- Return documents grouped in clusters to give you a sense of the major topics covered in the documents.
- Select a section of relevant text in a document and search for other, similar documents.
- Index many different document types, such as MicrosoftWord, and FrameMaker.
- Sort documents using up to 16 sort orders.
- Integrate backup and restore capabilities.
- Change the value of a configuration parameter using a system procedure.
- Optimize indexes for text searches when your server is inactive, to enhance performance.
- Create additional system management reports for viewing setup information.
- Ability to bring databases online automatically for text searches.

The Enhanced Full-Text Search product supports Sybase Failover. If an Sybase ASE fails, the Enhanced Full-Text Search accepts connections from the companion server. Additionally, if the Sybase ASE has proxy database support enabled, then both the primary and companion servers can use the Enhanced Full-Text Search at the same time.

11.1.11 HISTORICAL SERVER

This server obtains Sybase ASE performance data from Monitor Server and saves the data in files for deferred analysis. Historical Server is a Sybase Open Server application. The Historical Server control file maintains information about recording sessions that users create. This information persists across start-ups, so users can access recording sessions that they created during previous executions of Historical Server. The control file restricts user access to private recording session files. The current execution of Historical Server can access data files from previous executions only if the current execution is using the same control file, in the same home directory, as the previous executions were using. Therefore, in most cases, you should not change the Historical Server home directory between start-ups.

Historical Server concepts:

- Recording sessions
- Playback sessions
- Views
- Data items and statistic types

11.1.11.1 RECORDING SESSIONS

Recording sessions gather Sybase ASE performance data and store it in files for later analysis. Some attributes of a recording session are:

- **Monitor Server name** – by association, this defines the Sybase ASE whose performance you are monitoring.
- **Sample interval** – this attribute defines how often to collect performance data.
- **Views, alarms, and filters** – views and filters define the data you want to collect. Alarms define actions that can occur when a specified data item hits a predefined threshold value.
- **Start time and end time** – these specifications define the time period during which you want to collect the data.

When you create a recording session, Historical Server assigns it a session ID. You can list the session IDs of defined recording sessions using the **hs_list** command. **hs_list** can also show the complete recording session definition, including view names and the data items, alarms, and filters in the view.

Historical Server stores these recording session definitions in its control file, which resides in the Historical Server home directory. Therefore, **hs_list** can see only recording session definitions that were created by Historical Server instances using the same home directory that the current Historical Server is using.

To examine the data gathered by a recording session, you can:

- Populate Sybase ASE tables with the data from recording sessions by using the Sybase bulk copy (**bcp**) utility. See Appendix H for more information.
- Initiate a Historical Server playback session.

11.1.11.2 PLAYBACK SESSIONS

Playback sessions let you retrieve the data gathered during one or more recording sessions. You can play back data in two forms:

- **Playback to a client** – the results of the playback are sent to the user, who can view the results on the terminal or redirect them to a file.
- **Playback to a file** – the results of the playback are stored in a file. The resulting files are essentially a new recording session. You can use these files as input to yet another playback session, or as input to the **bcp** utility to populate Sybase ASE tables, or any other way that you would use recording session files.

The following attributes define a playback session:

- **Input recording sessions** – the input to a playback session is one or more recording sessions.
- **Views, start time, and end time** – these attributes define the data from the input recording sessions that you want to include in the playback session.
- **Summarization level** – you can specify raw playback, which shows you exactly what was recorded, or you can specify various summarization levels.

11.1.11.3 VIEWS

A recording session view defines the performance data you want Historical Server to gather. A playback session view defines which performance data from a recording session view you want Historical Server to play back.

A view consists of a view name and one or more data items. Each data has a statistic type associated with it.

When you define a recording session, you define one or more views to be included in that recording session. A recording session must have at least one view.

When you define a playback session, you define which views in the previously defined recording sessions should be included in the playback session. The playback session view names must be the same as the names used for the recording sessions views. You can include all data items or a subset of the data items from the recording session view in the corresponding playback view.

11.1.11.4 DATA ITEMS AND STATISTIC TYPES

A **data item** identifies specific information that you want to include in the view. If a data item includes embedded spaces, you must surround the name with quotation marks when you use it. Some sample data items are: Page I/O, Login Name, and CPU Time.

Each data item has a **statistic type** associated with it. The statistic type defines the duration of the data item (sample or session) and whether Historical Server performs calculations on the data item.

The statistic types contain embedded spaces. You must surround them with quotation marks when you use them in the Historical Server commands.

11.1.12 SCROLLABLE CURSORS

Sybase ASE allows both scrollable and non-scrollable cursors, which can be either semi-sensitive or insensitive. "Scrollable" means that you can scroll through the cursor result set by fetching any, or many, rows, rather than one row at a time; you can also scan the result set repeatedly. You must use Transact-SQL or JDBC to declare a scrollable cursor, and you must have the query engine provided in Sybase ASE 15.0 or later. A scrollable cursor allows you to set the position of the cursor anywhere in the cursor result set for as long as the cursor is open, by specifying the option first, last, absolute, next, prior, or relative in a fetch statement. To fetch the last row in a result set, enter:

```
fetch last [from] <cursor_name>
```

Or, to select a specific row in the result set, in this case the 500th row, enter:

```
fetch absolute 500 [from] <cursor_name>
```

"Insensitive" or "semi-sensitive" refers to the extent to which data changes from outside the cursor are visible to the cursor. A cursor can be semi-sensitive but not scrollable.

All scrollable cursors are read-only. All update cursors are non-scrollable.

11.1.13 AUTOMATIC UPDATE STATISTICS

Instead of manually running update statistics at a certain time, you can set update statistics to run automatically at the time that best suits your site and avoid running it at times that hamper your system. The best time for you to run update statistics is based on the feedback from the datachange function. Datachange also helps to ensure that you do not unnecessarily run update statistics. You can use these templates to determine the objects, schedules, priority, and datachange thresholds that trigger update statistics, which ensures that critical resources are used only when the query processor generates more efficient plans. Because it is a resource intensive task, the decision to run update statistics should be based on a specific set of criteria. Some of the key parameters that can help you determine a good time to run update statistics are:

- How much has the data characteristics changed since you last ran update statistics? This is known as the “datachange” parameter.
- Are there sufficient resources available to run update statistics? These include resources such as the number of idle cpu cycles and making sure that critical online activity does not occur during update statistics.

Datachange is a key metric that helps you measure the amount of altered data since you last ran update statistics, and is tracked by the datachange function.

Using this metric and the criteria for resource availability, you can automate the process of running update statistics. The Job Scheduler provides the mechanism to automatically run update statistics. Job Scheduler includes a set of customizable templates that determine when update statistics should be run. These inputs include all parameters to update statistics, the datachange threshold values, and the time when to run update statistics. The Job Scheduler runs update statistics at a low priority so it does not affect critical jobs that are running concurrently.

11.1.14 MULTIPLE TEMPORARY DATABASE

Sybase ASE allows you to create and manage multiple temporary databases in addition to the system tempdb database, which you can then use to create temporary objects such as private temporary tables and work tables. Database administrators can bind—that is, create associations between—the “sa” login and applications to specific temporary databases or to the *default* group of temporary databases using **sp_tempdb**. The *default* group is a system-created group that always has at least the system *tempdb* as its member. You can add other temporary databases to this group.

User-created temporary databases are created by the user, typically the database administrator. These databases are usually created to minimize resource contention (such as system catalog and log contention) in the system *tempdb*. User-created temporary databases are very similar to the system *tempdb* in that they are:

- Used primarily to create temporary objects
- Re-created, rather than recovered, during a system-recovery process

All objects in a temporary database before a shutdown or crash are lost during recovery because temporary databases are overwritten with the *model* database. Those restrictions that apply to the system *tempdb* also apply to the user-created temporary databases.

During login, sessions get assigned to a temporary database based on the existing bindings in effect:

- If the binding is to a specific temporary database that is online and available, the session gets assigned to it.
- If the binding is to the *default* group, a temporary database from that group is selected using a round-robin selection policy.
- If no binding is specified, a temporary database is selected from the *default* group.

The temporary database chosen for a session remains in effect for the duration of that session and never gets changed, regardless of any changes to the bindings.

Once a session is assigned a temporary database, all temporary objects created during that session are created in that temporary database. These objects are implicitly dropped when the session or server shuts down. Shareable temporary tables are implicitly dropped when the server shuts down.

11.2 MONITORING TOOLS

Sybase ASE provides the monitoring tools for monitoring the performance of the queries, monitoring concurrency and to monitoring overall ASE.

11.2.1 MONITORING THE PERFORMANCE OF THE QUERIES

The following set command can be used when testing the performance of individual queries. It will give information on the use of indexes, worktables and the number of physical and logical I/O's performed.

```
set showplan on
go
statistics io on
go
statistics time on
go
<query>
go
```

Other options are the use of traceflags (3604 or 3605) in combination with (302, 310). Trace flag 3605 sends the output to the Sybase errorlog, 3604 sends the output to the client so it can be captured into a file. Traceflag 302 and 310 give more information on the choice of indexes or table scan and the cost of the plan.

```
dbcc traceon (3604, 302, 310)
go
<query>
go
```

11.2.2 MONITOR CONCURRENCY

Use the system stored procedure sp_objectstats to check on any concurrency on the tables. This is especially important when operating with the tables in lockscheme "allpage".

11.2.3 MONITOR THE ASE SERVER

Regular monitoring of ASE activity and resource usage will minimize disaster recovery situations and allow for better upgrade planning. The troubleshooting activity will also prevent minor problems (blocking locks, deadlocks, hanging connections) from becoming major irritations.

The system stored procedure `sp_sysmon` is an invaluable tool when attempting to perform performance and tuning work in the ASE environment. The **`sp_sysmon`** stored procedure reports several hundred performance-related metrics about the server on which it is executed. It is normally executed for a specified interval (e.g., 5 minutes) while important or problematic workload is run on the server. With the **`begin_sample`** and **`end_sample`** parameters, you can invoke **`sp_sysmon`** to start sampling, issue queries, and end the sample and print the results at any point in time. The stored procedure monitors a large number of internal counters within the server and then produces a comprehensive report showing the information in a helpful, readable form.

Examples of the information provided in the `sp_sysmon` report include CPU utilization by CPU, worker processes used, parallel query processing performed, task context switching, priority queue use, transaction profiles, index management, lock management, cache management for different caches and cache types, and I/O statistics. Use of `sp_sysmon` provides detailed, non-intrusive, and accurate reporting that can help make it significantly easier to eliminate server performance problems.

11.2.3.1 ASE SERVER PERFORMANCE STATISTICS

Monitor server collects Sybase ASE performance data in real time and makes these statistics available to its clients, monitors in Sybase Central, Monitor Historical, and other applications written with Monitor Client Library.

Historical Server collects performance information from Monitor Server and saves the information in files for deferred analysis. Historical Server interfaces let users specify the data to collect and the time period desired.

System administrators can use this information to identify potential resource bottlenecks, to research current problems, and to tune for better performance.

Sybase ASE Monitor provides feedback for tuning at several levels:

- Sybase ASE configuration

- Database design
- SQL statements in applications and stored procedures

Note: The latest monitoring product in the Sybase family is Sybase Control Center. This tool allows monitoring and managing remote servers through a GUI interface. More details on this product in a separate chapter below.

11.2.3.2 SERVER ENTERPRISE MONITOR CLIENT LIBRARY

Monitor Client Library is an application-programming interface (API) for developing client applications that connect to Sybase ASE, Sybase ASE Monitor Server (Monitor Server), and Sybase ASE Historical Server (Historical Server) to gather performance data.

11.3 JOB SCHEDULER

Job Scheduler eases the management of Sybase ASE by providing the ability to define and schedule database administration tasks. With Job Scheduler, jobs that normally require interaction from a database administrator can be scheduled to run unattended at the appropriate times, freeing the database administrator to attend to other issues.

Job Scheduler 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 Sybase ASE task
- An external process called the JS Agent
- The *sybmgtadb* 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 *sybmgtadb*. 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 *sybmgtadb* database.

All the job, schedule, and scheduled job information, and data needed by the JS task for internal processing is stored in the *sybmgtadb* database. Most access to data in the *sybmgtadb* 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 *sybmgtadb* 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.

11.4 SYBASE CONTROL CENTER

Sybase Control Center is a server application that uses a Web-browser-based client to deliver an integrated solution for monitoring and managing Sybase products.

Sybase Control Center provides a single comprehensive Web administration console for real-time performance, status, and availability monitoring of large-scale Sybase enterprise servers. Sybase Control Center combines a modular architecture, a rich client administrative console, agents, common services, and tools for managing and controlling Sybase products. It includes historical monitoring, threshold-based alerts and notifications, alert-based script execution, and intelligent tools for identifying performance and usage trends.

A Sybase Control Center server can support:

- Up to 50 monitored resources (servers)
- Up to 10 users logged in simultaneously

Sybase Control Center supports Sybase ASE Enterprise version 15.0.2 and later. It supports clustered configurations on Sybase ASE Cluster Edition version 15.0.3 and later.

The Sybase Control Center client/server architecture allows multiple clients to monitor all Sybase ASE Servers in an enterprise through a small number of Sybase Control Center servers. Sybase Control Center provides availability monitoring, historical performance monitoring, and administration capabilities in a scalable Web application that is integrated with management modules for other Sybase products. It offers shared, consolidated management of heterogeneous resources from any location, real time notification of availability and performance, and intelligent tools for spotting performance and usage trends, all via a thin-client, rich Internet application (RIA) delivered through your Web browser.

Use Sybase Control Center to track a variety of performance metrics, gathering statistics that over time will give you powerful insight into patterns of use and the behavior of databases, devices, caches, and processes on your servers. You can display collected data as tables or graphs. By plotting results over any period of time you choose, from a minute to a year, you can both see the big picture and focus on the particulars. Detailed knowledge of how your servers have performed in the past helps you ensure that Sybase ASE meets your needs in the future.

Each Sybase ASE server needs to have the Sybase Unified Agent configured and running. This is the method on how Sybase Control Center communicates with the Sybase ASE server. This enables you to remotely startup and shutdown the Sybase ASE server.

11.4.1 ALERTS

You can configure Sybase Control Center to notify you when a resource requires attention.

You do this by setting up a predefined alert that is triggered when a performance counter enters a particular state or passes a threshold value that you set. When the alert goes off, it generates an alert notification.

An alert notification takes the form of a visual indicator in the Alert Monitor and, optionally, an e-mail message. The Alert Monitor displays information about the alert, including the resource name, alert severity, value, and date. You can resolve the alert or allow it to escalate.

Configure, monitor, and control alerts for managed resources by:

- Enabling and disabling alert subscriptions for resources
- Configuring shell scripts to run when alerts fire
- Setting alert state or threshold triggers
- Responding to an alert by resolving it, adding notes if desired
- Modifying or deleting alerts
- Viewing alert history

11.4.2 REPOSITORY

The Sybase Control Center embedded repository stores information related to managed resources, as well as user preference data, operational data, and statistics.

You can back up the repository database on demand, schedule automatic backups, restore the repository from backups, and configure repository purging options. Full and incremental backups are available. A full backup copies the entire repository. An incremental backup copies the transaction log, capturing any changes since the last full or incremental backup.

By default, Sybase Control Center saves backups as follows:

- Each full backup is stored in its own subdirectory in <SCC-install-directory>/backup.
- Each incremental backup is stored in a file in <SCC-install-directory>/backup/incremental.

Sybase recommends that you periodically move backup files to a secondary storage location to prevent the installation directory from becoming too large.

11.4.3 LOGGING

A log is a record of events related to a server or a client.

In Sybase Control Center, logging helps system administrators identify errors and other system events by recording messages about the events in log files. Sybase Control Center maintains these logs:

- The client log—captures messages about activities in the browser-based client components. These messages are generated by the component products to display information that is pertinent to the user but not critical enough to warrant a pop-up. Sybase also uses the client log to trace client browser operations.
- Server logs—capture messages about activities during the initialization sequence, such as starting services; auditing messages recording logins and logouts; errors such as missed scheduled events; and other events on the server. Server logs include:
 - Component logs, which record only events concerning individual product modules
 - The agent log, a composite log that records events in all components and in the Sybase Control Center framework
- The repository log—captures information about inserts and updates that have occurred in the Sybase Control Center repository, a SQL Anywhere database.

11.4.4 HEAT CHART

The heat chart displays status and availability statistics for managed resources in the current perspective.

The heat chart displays the state of resources in your perspective—whether the resources are running, suspended, or down. In addition, the heat chart lists the type of each resource and provides statistical data, including the start time of the last data collection.

In the Perspective Heat Chart view, you can filter the resources that you want to see. You can also search and sort the results by column. From within the Perspective Heat Chart, you can right-click a resource to see a menu of monitoring and administrative options that vary based on the resource type.

Heat chart data is collected directly from managed servers, tagged with the date and time when it was collected, and stored in the Sybase Control Center repository.

11.4.5 HISTORICAL PERFORMANCE MONITORING

Monitor performance data to determine whether your environment is working efficiently.

Obtain detailed information about the status of the resources in your environment. You can create performance graphs that illustrate resource performance over a specified period of time.

11.4.6 MANAGE AND MONITOR A SYBASE ASE SERVER ENVIRONMENT

Monitor the performance, processes, databases, and other aspects of Sybase ASE.

- Clusters
 - Monitor Sybase ASE cluster configurations. Monitored resources include memory usage, device I/O, engine CPU utilization, connection information, inter process communication, workload management, load profiles, routes, and so on.
- Displaying the Performance Overview
 - The Overview screen shows Sybase ASE performance status.
- Caches
 - Monitor Sybase ASE data, procedure, statement caches, and in-memory storage.
- Databases
 - Monitor Sybase ASE databases.
- Devices
 - Monitor the devices used by Sybase ASE.
- Engines
 - Monitor Sybase ASE engines.
- Processes
 - Monitor Sybase ASE processes.
- Segments
 - Monitor the segments used by Sybase ASE databases.
- Transactions
 - Monitor active Sybase ASE transactions.
- Server Configuration Values
 - Monitor Sybase ASE configuration values.
- SQL Activity
 - Monitor SQL queries on Sybase ASE.
- Replication Agents

- Monitor RepAgent threads for replicate databases on the selected Sybase ASE.
- Settings
 - Learn about Sybase ASE monitoring controls on the Settings screen.
- Statistics
 - Availability and performance statistics in Sybase Control can help you identify if your system is running as efficiently as possible.

12 APPENDIX A – PORTABILITY CHECK

Migrating an Oracle database to Sybase ASE is a potential very large project and requires close attention to detail. So far this guide outlined the key differences between Oracle and Sybase ASE as well as provided workaround suggestions for more challenging migration tasks that do not offer a one-to-one migration path.

To summarize the workaround solutions the table below lists the Oracle functions, SQL code and/or architectural challenge that need special attention and cannot be migrated without changing the code logic.

12.1 WORKAROUNDS FOR SYBASE ASE

Oracle	Sybase ASE Workaround Approach
BEFORE STATEMENT Triggers	Creating a set of rules on the table column will emulate some of the trigger functionalities. To implement BEFORE trigger actions that perform a database lookup to validate data, a custom Sybase function has to be written and then referenced by the rule.
BEFORE ROW Triggers	This functionality can be implemented by using the pseudo tables inserted and deleted inside a T-SQL trigger. The inserted table contains the new values and the deleted table contains the existing values of a table row.
AFTER ROW Triggers	
Synonyms	Table and view synonyms can be implemented via a view in a separate database to cross reference the physical tables and views in the application database. This allows for security settings as well as using the short object name in the SQL statements. All other references to synonyms must be replaced with the fully qualified name to an object. <remote_server>.<database_name>.<owner>.<object_name>

Oracle

**Sybase ASE
Workaround Approach**

Sequences

For unique keys in a table sequences can be implemented with a identity column as primary key. To implement a sequence that is used as a running key sequence the following code example will help:

12.1.1.1.1 ORACLE CODE:

```
CREATE SEQUENCE test_seq
MINVALUE 1
STARTWITH 1
INCREMENTED by 1
CACHE 20;

INSERT INTO m_table VALUES (test.seq.nextval,..);
```

12.1.1.1.2 EQUIVALENT SYBASE CODE:

```
// create table
CREATE TABLE my_seq (seq int)
go

//initialize the sequence
INSERT INTO my_seq select 0
go

/*create stored procedure to get the incremented
value and to set the
incremental value */

CREATE PROCEDURE get_seq (@seq int OUTPUT)
AS
UPDATE my_seq SET seq = seq+1
SELECT @seq = seq FROM my_seq
go

// execute the sp to get the next sequence number
DECLARE @seq int
EXEC get_seq @seq OUTPUT
INSERT INTO m_table VALUES (@seq,..)

Go
```

Materialized Views

Sybase ASE does not support the concept of materialized views.

Oracle uses materialized views for one way replication via dblink connections and updateable materialized views for two-way replication.

Sybase Replication Server will handle these replication needs outside Sybase ASE.

Oracle

**Sybase ASE
Workaround Approach**

ANSI transaction processing

To enable ANSI transaction processing the following command has to be executed in every session.

```
set transaction isolation level 3
```

To avoid code rewrites, this can be done through a login trigger.

In addition all stored procedure must be set to “chained” operation mode.

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

The downside of this method is that all transactions will now be executed in a serial fashion instead of parallel execution. If the database server hardware is fast enough and not many transactions are being executed in parallel to begin with, this might be a viable solution. In heavy parallel environments, this will end up in unacceptable performance situations where client applications are waiting too long for locks to free up. The only way around this is to rewrite the transaction logic to conform to the T-SQL standard and splitting long transactions into smaller pieces.

Missing index types and table types

In Oracle there are many more table types and index types available to manage data placement and data access performance. Although this is common practice in Oracle and allows DBAs fine-tuned control of the data schemas and the data. This might not translate into the same requirements on Sybase ASE. Sybase ASE offers plenty of options to take control over data schemas, object placements and data access performance. It is the duty of the DBA to choose the right method for each occurrence.

Oracle

**Sybase ASE
Workaround Approach**

Missing data types or data type limits

There are a couple of data types in Oracle that do not translate into any Sybase ASE data type:

- CLOB
- BLOB

To avoid data type limits with VARCHAR2 the Sybase ASE server should be created with at least 8k page blocks.

BLOB and CLOB data should be closely evaluated and if there is a distinct need that will exceed the limits of the Sybase ASE data type image, then the use of Sybase IQ with its BLOB capabilities needs to be considered. This will also require additional application and architectural changes to synchronize data between Sybase ASE and Sybase IQ as well as a change in data access methods. Again, this needs to be closely evaluated.

Flashback

Sybase ASE does not provide the Flashback functionality or an equivalent. If you want to implement this feature you need to use a combination of two features from Sybase ASE.

Install and enable the auditing feature in Sybase ASE. This will allow for timestamp searches on actions that need to be reversed or data needs to be extracted before a certain action.

With the exact timestamp you will be able to create an archive database from backup logs and transaction logs.

1. Create an archive database based on the last full backup taken from this database.
 2. Apply all transaction logs to roll forward all transactions to the desired timestamp.
 3. Use the `until_time` parameter in the load transaction command on the last transaction log file to pinpoint the restore timestamp just before the desired timestamp. This will give you the exact data constellation before a destructive or faulty action has been taken place.
-

Oracle

**Sybase ASE
Workaround Approach**

Oracle HINTS

You must remove all Oracle HINTS from the SQL statements. You find a list of Oracle HINTS later in this section under the header SQL Hints. There is also a small SQL statement listed that will search the Oracle repository for any occurrence of a HINT keywords within stored procedures, functions and triggers.

Oracle ROWID

Every Oracle table contains a unique identifier that is always there, even if you don't have a primary key for the table defined.

To achieve a similar effect you can add an identity column to each tables named ROWID. This will avoid changes to the application code using ROWID to access records, most likely within cursor loops. Make sure that you use the upper case name, because Oracle is not case sensitive and the code will most likely be in upper case. You can only have one identity column per table. If there is already an identity column for the primary key, you need to create a column that receives the key from a key value table (Sybase's version of a sequence) and manage the data via an INSERT trigger. This method allows for a lower case and an upper case version of ROWID to cover all implementations of this method, even in embedded application SQL code outside the database server.

Oracle

**Sybase ASE
Workaround Approach**

Oracle ROWNUM

Sybase ASE does not implement pseudo columns with its table store. The Oracle pseudo column ROWNUM does not exist, but comparable functionality can be achieved.

Within a cursor after every FETCH command, the global variable @@rowcount can be accessed and stored into a temporary table for further processing. The @@rowcount starts at 0 when the cursor opens the first time. This becomes more complicated when using scrollable cursors. @@rowcount simply increases the value by 1 every time a FETCH command gets executed. If the fetch cursor moves back and forth inside the results set, @@rowcount will get out of sync.

ROWNUM must be removed from the SQL. If the ROWNUM is part of the SARG, the following has to be implemented.

If ROWNUM is part of the program logic, set the max number of rows before the SQL executes and then reset it.

Oracle:

```
SELECT * FROM emp WHERE state = 'CA' AND ROWNUM < 21
```

Sybase:

```
Set rowcount 20
```

```
SELECT * FROM emp WHERE state = 'CA'
```

```
Set rowcount
```

Oracle

**Sybase ASE
Workaround Approach**

Oracle Resource Manager

Sybase ASE provides the Resource Governor to set limits on how resources are being used by the Sybase ASE database and its processes.

The Oracle Resource Manager triggers actions based on CPU consumption and I/O. Sybase ASE can match this and add numbers of rows selected and the amount of space used in the tempdb as triggers for actions.

The biggest difference is what actions will be implemented when resource limits are being reached. Oracle maintains rules based action diagrams that will reduce the available resources for the offender and moves sessions into a lower resource pool.

Sybase ASE does not throttle sessions that reaches the resource limits. It sends a warning, aborts the query / transaction or ends the user session.

There is no workaround for this action.

Oracle SQL Plan Management

Oracle has the ability to store and maintain query plans to support the query optimizer to make better decisions. Every time a query gets execute, the query optimizer compares the current query plan with the stored query plan and choses the better plan automatically.

Sybase ASE can create abstract query plans that then can be manually attached to a query. This is mostly used to reproduce production query behavior in a test environment where the data volume would not allow the query optimizer to make the same exact decisions.

Sybase ASE does not provide a similar functionality to the Oracle SQL Plan Management.

Oracle	Sybase ASE Workaround Approach
AWR (Automatic Workload Repository)	<p>When activated, Oracle stores every query with corresponding performance indicators and metrics in a repository called AWR (Automatic Workload Repository). By default, 7 days worth of queries will be captured and stored.</p> <p>Queries can be analyzed and it allows to report the top poorly performing queries automatically to be tuned and new query plans managed by the SQL Plan Management will be used when the query are being executed again.</p> <p>With Sybase ASE a similar repository can be created using MDA tables and a custom developed data storage function that exports the MDA table information on a regular interval, 1 minute or less, and stores the information in a repository for future reference and analysis. These analytical reports have to be custom created as well.</p> <p>Sybase ASE does not provide the data export or the reporting functionalities.</p>

12.2 ORACLE TO SYBASE ASE INCOMPATIBILITIES

Sometimes there are situations where a direct migration from Oracle to Sybase ASE is not possible due to incompatible functionalities or architectural incompatibilities. It is important to know about possible incompatibilities **before** you start a migration project.

The following list contains incompatible functionalities, types, pseudo columns and commands

- **Object Tables**

These are tables created on the user defined type OBJECT. As an example:

```
CREATE OR REPLACE TYPE person_typ AS OBJECT (  
  ssn NUMBER(9), name VARCHAR2(30), address VARCHAR2(100))  
NOT FINAL;  
/  
  
CREATE TABLE person_obj_table OF person_typ;
```

```
INSERT INTO person_obj_table
VALUES (person_typ(20, 'Bob Jones', '111-555-1212'));

set linesize 121
col address format a30

SELECT * FROM person_obj_table;

SELECT p.object_id, p.object_value FROM person_obj_table p;
```



To check for this occurrence scan the Oracle SQL code for the pseudo columns OBJECT_ID and OBJECT_VALUE.



Sybase ASE can define Java classes as column data types. This does not offer entire tables as objects, but allows you to store objects as columns in a table. This may trigger a rewrite of the application code that uses object tables. A possible workaround is to create a table for each object table and move the object storage from the table level to the column level with a column with a Java class datatype.

- **Pseudo Columns**

Not supported: ORA_ROWSCN, VERSION_XID, VERSION_STARTSCN, VERSION_ENDSCN, VERSION_STARTTIME, VERSION_ENDTIME, VERSION_OPERATION

- **SQL Syntax**

The following SELECT syntax attributes are not supported by Sybase ASE. This may trigger a complete rewrite of the application logic.

- ONLY
- INTERSECT
- MINUS
- FOR
- UNPIVOT
- INCLUDE
- EXCLUDE
- NULLS

- SAMPLE
 - SEED
 - CROSS
 - NATURAL
 - CONNECT BY
 - NOCYCLE
 - START WITH
 - ROLLUP
 - CUBE
 - GROUPING SETS
 - MODEL
 - IGNORE
 - KEEP
 - NAV
 - UNIQUE
 - DIMENSION
 - SINGLE REFERENCE
 - RETURN ALL ROWS
 - RETURN UPDATED ROWS
 - REFERENCE
 - ON
 - MAIN
 - PARTITION BY
 - DIMENSION BY
 - MEASURES
 - RULES
 - ITERATE
 - SIBLINGS
 - NULLS FIRST
 - NULLS LAST
 - GROUPING SETS
 - NOWAIT
 - WAIT
 - SKIP LOCKED
 - AS OF TIMESTAMP
 - SYSTIMESTAMP
 - AS OF
- **Oracle Web Access from PL/SQL**

The function calls from PL/SQL against the Oracle Web Services are incompatible with Sybase ASE.

Check the PL/SQL code for the following keywords:

- OWA_CUSTOM
- OWA_CX
- OWA_OPT_LOCK
- OWA_SEC
- OWA_TEXT
- OWA_UTIL

- **SQL*Plus**

The Sybase ASE counterpart for SQL*Plus is isql. Unlike isql, where all settings are being passed in as a command parameter, SQL*Plus allows for more complex configuration settings inside SQL*Plus. In order for SQL and PL/SQL code from script files be able to run inside isql, these configurations must be removed.

This can alter the output of the code and may change the application logic.

12.3 SEARCH FOR KEYWORDS IN ORACLE SOURCE

Before you can migrate an Oracle schema or Oracle stored procedure, functions and triggers, you need to check for keywords that we already identified as either problematic or non-migratable.

The following code will scan any object within the Oracle database for certain keywords and returns the name and owner of the object as well as the object type:

```
SELECT owner, name, type FROM dba_source WHERE instr(UPPER(text), UPPER('<keyword>'))  
> 0
```

Sometimes it is important to retrieve the exact code and line number of the occurrence within a stored procedure, function or trigger. Please be cautious while using this option as it can return a lot of data:

```
SELECT owner, name, type, line, text FROM dba_source WHERE instr(UPPER(text),  
UPPER('<keyword>')) > 0
```

13 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.

Planning Migration:

- ✓ Use Appendix A to check for migration incompatibilities
- ✓ Use the guide to identify problem points and develop workarounds to emulate certain functionality
- ✓ Ensure all migration challenges are addressed and mitigating actions are planned for

Architecture Migration:

- ✓ Creation of Sybase ASE 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
- ✓ Set the appropriate transaction isolation level
- ✓ Creation of security – logins, roles and permissions

Schema Migration:

- ✓ Reverse engineering of all schemas
 - Use PowerDesigner to convert data types
- ✓ Plan the object placement
- ✓ Create the new index strategy
- ✓ Migrate schema security and rules

Data Migration:

- ✓ Migration of data
 - Use bcp for large data volumes
 - Use of ECDA and Sybase Replication Server for up-to-date data movement
- ✓ 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:

- ✓ Address the transaction processing method change from ANSI to T-SQL
- ✓ 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

Validate Migration:

- ✓ Use sp_sysmon and sp_objectstats to check for any system bottlenecks
- ✓ Count the number objects in both the database – Oracle and Sybase ASE
- ✓ Count the number of rows of each table in both the database
- ✓ Run the application under real world user load to simulate potential blocking contentions that could create a performance issue with transactions.

Use this checklist to develop a more detailed checklist for your application and database. It is absolutely crucial that you understand where the problem points are and how to deal with them. This

guide provides many alternative solutions for migration challenges, but it is impossible to predict all situations you may encounter. By applying the same strategies to mitigate migration challenges as outlined in this guide, you will be well equipped to handle any challenge.

14 APPENDIX C – DATA TYPE DIFFERENCES

Oracle	Sybase ASE	Comments
VARCHAR2(n)	VARCHAR(n)	
NVARCHAR2(n)	NVARCHAR(n)	
CHAR	CHAR	
CHAR(1)	BIT	true/false and yes/no types of data can be stored. Storage size is 1 byte, can hold either 0 or 1
CHAR(n)	CHAR(n)	
NCHAR(n)	NCHAR(n)	
NUMBER(3)	TINYINT	Sybase ASE tinyint ranges from 0-255
NUMBER(4)	SMALLINT	Sybase ASE smallint ranges from –32768 to +32767
NUMBER(9)	INT	Range of integer is 231 -1 (2,147,483,647) to -231 (-2,147,483,648)
NUMBER(19,4)	MONEY	money and smallmoney datatypes stores monetary data. It stores to 4 digits of precision to the right of the decimal point, and to approx +/- \$214k for smallmoney, approx +/- \$922T for money to the left of the decimal.
NUMBER(19,4)	SMALLMONEY	See above comments for Sybase ASE money
NUMBER	REAL, FLOAT, DOUBLE	
FLOAT	FLOAT, REAL	
FLOAT(p)	DOUBLE PRECISION	
FLOAT(p)	FLOAT(p)	
NUMBER(p,s)	NUMERIC(p,s)	The number(p,s) datatype of oracle map exactly to numeric and decimal types of Sybase ASE
NUMBER(p,s)	DECIMAL(p,s)	See above comments Oracle number
BINARY_FLOAT	-	
BINARY_DOUBLE	-	
DATE	DATETIME	datetime values are accurate to 1/300 second on platforms that support this level of granularity, holds

Oracle	Sybase ASE	Comments
		dates between January 1, 1753 and December 31, 9999
TIMESTAMP	BIGDATETIME	
DATE	DATE	holds dates from January 1, 0001 to December 31, 9999. Storage size is 4 bytes
DATE	SMALLDATETIME	holds dates from January 1, 1900 to June 6, 2079, with accuracy to the minute. Its storage size is 4 bytes
	TIME	time is between 00:00:00:000 and 23:59:59:999
TIMESTAMP	BIGDATETIME	datetime value is accurate to 1/300 second
CLOB	TEXT	text are variable-length columns that can hold up to 2,147,483,647 (231 - 1) bytes of printable characters.
BLOB	IMAGE	image are variable-length columns that can hold up to 2,147,483,647 (231 - 1) bytes of bytes of raw binary data.
LONG	TEXT	text are variable-length columns that can hold up to 2,147,483,647 (231 - 1) bytes of printable characters.
RAW(n)	BINARY(n)	
RAW(n)	VARBINARY(n)	
RAW(n)	IMAGE	
BFILE	-	Not supported

15 APPENDIX D – SQL LANGUAGE DIFFERENCES

Following is a table and examples of differences between the SQL:

S_R stands for Search and Replace.

15.1 GENERAL

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
1	/	GO This "slash" is contained at the end of some of the procedures examined. It is the equivalent to the Sybase ASE GO command at the end of a stored procedure coded to run via an ISQL command.	Search and Replace (S_R) / as GO S_R
2	Semicolon Use in Oracle SQL	Global replace with a space.	S_R selective
3	Variable declaration e.g x:= expression;	SELECT @x = expression	
4	Parameter Naming Convention	Translate in Sybase ASE to preceding character @. May cause "length" problems of existing 30 character long variable names in existing Oracle code.	S_R and manual recode.
5	select INTO variable	This is the Oracle convention for transferring data value to a variable. Use the Sybase ASE syntax - select @variable1=col1, @variable2=col2... ad inf.	Manual
6	CONSTANTS		Redefine as variables and check scope of usage (local or global).
7	%TYPE	Code manual field definition to replace this.	Look up Oracle field datatype definition and code exact specification.

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
8	%ROWTYPE	Code each field as an individual variable.	Manual recode.
9	IS RECORD OF	Code each field as an individual variable.	Manual coding conversion.
10	IS TABLE OF	Code as temp table and use FETCH or SELECT to process row.	Redesign code to temp table with cursor scan processing.
11	dbms Oracle calls	Recode to Sybase ASE functions such as RAISERROR or PRINT and other Sybase ASE System Stored Procedures.	Manual recode.
12	Dynamic SQL	Recode to Sybase ASE T-SQL. PowerBuilder can issue SQL dynamically, but Sybase ASE does not support robust dynamic SQL processing.	Manual recode.
13	Using DFAULT E.g. blood_type char := 'O'	Use "=" sign E.g. @blood_group char = 'O'	
14	EXIT WHEN nDbcdDone = 1 AND nAWMSDone = 1;	Convert this to an expression and a T-SQL BREAK or RETURN statement as a part of a conditional IF statement.	Manual recode.
15	INITCAP	Translate to Sybase ASE stored procedures or Open Client / Open Server applications.	

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
16	<p>SEQUENCE or CREATE SEQUENCE</p> <p>E.g CREATE SEQUENCE test_seq MINVALUE 1 STARTWITH 1 INCREMENTED by 1 CACHE 20;</p> <p>INSERT INTO m_table VALUES (test.seq.nextval,...);</p>	<p>A key table will be set up and stored procedures defined to emulate the NEXTVAL function in Oracle. The specs of this is as follows:</p> <pre>CREATE TABLE my_seq (seq int) go INSERT INTO my_seq select 0 go CREATE PROCEDURE get_seq (@seq int OUTPUT) AS UPDATE my_seq SET seq = seq+1 SELECT @seq = seq FROM my_seq go DECLARE @seq int EXEC get_seq @seq OUTPUT INSERT INTO m_table VALUES (@seq,...) Go</pre>	Manual recode. Will involve schema changes.
17	synonym	Recode manually to the database.table.column_name, which is to be referenced for each field reference.	Manual recode.
18	TYPE AccountingRec IS RECORD	Code each field as an individual variable.	Manual recode.

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
19	UNION	Evaluate UNIONS in CURSORS and recode. Note that you cannot use a UNION command in the SELECT statement of an updatable cursor. There are also special conditions for same number of expressions in tables of the UNION, and the order in which the columns must appear.	Manual recode.
20	userenv('sessionid')	The kernel pid from master..sysprocesses identifies the session's unique thread id. Use the following syntax as an equivalent: select kpid from master..sysprocesses where status = "running".	Manual recode.
21	VARIABLE HANDLING - GLOBAL AND LOCAL	Local variables may be used in a procedure and then their value passed onto another called or sub-procedure or back to a calling procedure. May involve special coding for any Global variables or variables used in a package which was converted into separate stored procedures.	Manual recode.
22	user-defined functions	Use Sybase ASE Stored Procedures to pass data to; to operate on it; and return it afterwards. The EXECUTE command is used to run a system procedure or a user-defined stored procedure.	Manual recode.
23	IS	In procedure declaration, translate to AS	S_R selective.

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
24	NUMBER type in stored procedures	use NUMERIC(10)	
25	Packages and Procedures Private and Public	Translate packages to single stored procedures where possible, and split into separate stored procedures where necessary.	Manual recode.
26	Triggers (BEFORE TRIGGERS)	RULES can be used in Sybase ASE to specify the domain of acceptable values for a particular column, or for any column of a user-defined datatype. Also evaluate each condition and possibly use CHECK integrity constraints with the CREATE TABLE statement.	Manual recode.
27	RETURN BOOLEAN	Translate to return an integer expression and evaluate in receiving procedure or statement. Since Boolean is not an integer_expression, it is not allowed as a parameter to the Sybase ASE RETURN parameter. This will need to be translated to the integer expressions (0,1) and translated when received for processing. 1=False; 0=True. Instead of using the numeric literals 1 or 0, the two variables should be defined as follows: @false integer=1 and @true integer=0; so that the stored procedure should return either @false or @true.	Manual recode.

15.2 OPERATORS

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
1	Concatenation operator 	+ Sybase ASE supports a string concatenation operator. It is the plus + sign.	S_R as +
2	Modulo operator MOD	%	S_R
3	Comparison Operator IF rAcct.AuthNum IS NULL THENBEGIN..END E.g IF VAR IS NULL Or IF VAR IS NOT NULL	Evaluate coding and translate NULL to IS NULL or NOT NULL where necessary. May involve definition of columns at schema level. E.g. IF @VAR = NULL Or IF @VAR != NULL	Manual recode and possible schema alterations.

15.3 FUNCTIONS

15.3.1 SYSTEM FUNCTIONS

	Oracle	Sybase ASE	Conversion
#	Code	Translation	Activity
1	CEIL Returns the smallest integer greater than or equal to the specified value	CEILING	S_R
2	TRUNC String functions	CONVERT(INT,..)	S_R
3	SUBSTR()	SUBSTRING()	S_R
4	LENGTH()	CHAR_LENGTH()	S_R
5	CHR()	CHR()	S_R
6	TO_CHAR	CONVERT	Manual recode.
7	TO_NUMBER	CONVERT(NUMERIC(n,n),field)	Manual recode.
8	convert_char_to_number	This is a custom-defined character to numeric conversion routine. It can be replaced with the Sybase ASE function (convert(decimal(x,y), "string"). For the purposes of the s2k applications set x to 18, and y to 9.	S/R &Manual recode.

15.3.2 DATA FUNCTIONS

	Oracle	Sybase ASE	Conversion
#	Code	Translation	Activity
1	DATE functions and calculations	Recode to Sybase ASE date handling routines. Sybase ASE supports a large selection of date and time functions such as DATEADD, DATEDIFF, DATEPART(), DATENAME(), timestamp and others.	S_R and manual recode.

	Oracle	Sybase ASE	Conversion
#	Code	Translation	Activity
2	LAST_DAY	Translate to Sybase ASE stored procedures or Open Client / Open Server applications.	Manual recode.
3	SYSDATE, SYSTIMESTAMP	Use Sybase ASE GETDATE() and proper formatting functions.	S_R selective.
4	DATE and TIMESTAMP Functions	Replace in some cases using a local variable, or it can be re-architected to use the Sybase ASE User Defined TIMESTAMP data type.	Manual recode.
5	TRUNC(dstartdate)	Use Sybase ASE CONVERT and the format that returns the date at 00:00 hour.	Manual recode.

15.3.3 CONDITIONAL FUNCTIONS

	Oracle	Sybase ASE	Conversion
#	Code	Translation	Activity
14	NVL	Translate NVL to ISNULL.	S_R selective.
15	NVL2 E.g NVL2(salary,salary*2,0)	CASE structure E.g. CASE WHEN salary = NULL THEN 0 ELSE salary * 2 END	

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
16	DECODE Used to evaluate the values with 'if-else' logic	<p>Convert to Sybase ASE case expression code.</p> <p>For e.g.</p> <pre> SELECT STUDENT_ID, (CASE WHEN COURSE_ID = 101 THEN 1 ELSE 0 END) AS COURSE_101, (CASE WHEN COURSE_ID = 105 THEN 1 ELSE 0 END) AS COURSE_105, (CASE WHEN COURSE_ID = 201 THEN 1 ELSE 0 END) AS COURSE_201, (CASE WHEN COURSE_ID = 210 THEN 1 ELSE 0 END) AS COURSE_210, (CASE WHEN COURSE_ID = 300 THEN 1 ELSE 0 END) AS COURSE_300 GROUP BY STUDENT_ID ORDER BY STUDENT_ID </pre>	Manual recode

15.4 JOINS

	Oracle	Sybase ASE	Conversion
#	Code	Translation	Activity
1	col1 = col2(+) or col1(+) = col2	Translates to Sybase ASE T-SQL as col1 =* col2 or col2 *= col1)	S_R

15.5 CURSORS

	Oracle	Sybase ASE	Conversion
#	Code	Translation	Activity
1	CURSOR usage	Change CURSOR definition text to Sybase ASE text.	Recode.
2	CURSOR processing options %ISOPEN, %NOTFOUND, %FOUND, %ROWCOUNT	Use @@rowcount and @@sqlstatus Sybase ASE global variables.	Recode.

15.6 CONTROL OF FLOW LANGUAGE

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
1	IF expression THEN statement; END IF;	IF expression statement The IF or ELSE keywords affect the performance of only a single SQL statement unless statements are grouped into a block between the keywords BEGIN and END. Recode to BEGIN - END or eliminate END IF; where appropriate.	Evaluate and manual recode.
2	EXIT;	BREAK	S_R
3	LOOP and END LOOP	Translate to WHILE, CONTINUE, GOTO and IF ELSE T-SQL statements and procedures.	Manual recode.

15.7 ERROR HANDLING

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
1	SQLCODE	@@error	S_R
2	Generic Error Handling	Detect condition by evaluating Sybase ASE system functions as defined above. Add coding for exception or error handling.	Manual recode.
3	WHEN TOO_MANY_ROWS THEN (Error Handling)	Detect TOO_MANY_ROWS by evaluating Sybase ASE @@rowcount. Add coding for error handling	Manual recode.
4	WHEN NO_DATA_FOUND	Detect by evaluating Sybase ASE @@rowcount. Add coding for error handling.	Manual recode.
5	Error handling such as EXCEPTION	Recode to Sybase ASE error and message handling.	Manual recode.
6	Oracle use of cursor_already_open, dup_val_on_index, invalid_cursor, invlaid_number, login_denied, no_data_found, not_logged_on, program_error, storage_error, timeout_on_resource, too_many_rows, transaction_backed_out, value_error, zero_divide.	Detect condition by evaluating Sybase ASE system functions as defined above. Add coding for exception or error handling.	Manual recode.
7	WHEN DUP_VAL_ON_INDEX THEN EXCEPTION	Evaluate Sybase ASE error code returned from the INSERT or other command, which caused this error. Add coding for exception or error handling.	Manual recode.

Oracle		Sybase ASE	Conversion
#	Code	Translation	Activity
8	PRAGMA and other Oracle compiler directives	Detect condition by evaluating Sybase ASE system functions as defined above. Add coding for exception or error handling.	Manual recode.
9	RAISE_APPLICATION_ERROR (-20300, 'Task has no Reason Code supplied');	Error handling in Sybase ASE can be performed in a number of ways. The simplest way is to use the RAISERROR or PRINT commands to issue or print a text message. Another method is to create system messages and link them with constraint violations or conditions.	

15.8 EXAMPLES OF THE DIFFERENCES IN SQL

15.8.1 STRING MANIPULATION

Oracle	Sybase ASE
<code>SUBSTR(phone,1,3) AS AREA_CODE</code>	<code>SUBSTRING(phone,1,3) AS AREA_CODE</code>
<code>(first_name ' ' mi_name ' ' last_name) as name</code>	<code>(first_name + ' ' + mi_name + ' ' + last_name) as name</code>

15.8.2 NUMERIC MANIPULATION

Oracle	Sybase ASE
<code>SELECT student_id = '100'</code> (acceptable, but not a good idea)	<code>SELECT student_id = 100</code>
<code>LTRIM(RTRIM(TO_CHAR (course_ceu, '999D99')))</code>	<code>RTRIM(CONVERT(CHAR,ROUND(course_ceu,2)))</code>
<code>LTRIM(RTRIM(TO_CHAR(COURSE_CEU, '999')))</code>	<code>RTRIM(CONVERT(CHAR,CONVERT(INT, COURSE_CEU)))</code> (not ROUND)

15.8.3 CONTROL STRUCTURES

Oracle	Sybase ASE
<code>LOOP</code> <code>monthly_value := daily_value * 31;</code> <code>EXIT WHEN monthly_value > 4000;</code> <code>END LOOP;</code>	<code>WHILE @monthly_value <= 4000</code> <code> SELECT @monthly_value =</code> <code> @daily_value * 31</code>

Oracle	Sybase ASE
<pre>FOR i IN 1..5 ... END LOOP;</pre>	<pre>DECLARE @lb int, @ub int, @i SELECT @lb = 1, @i = @lb, @ub = 5 WHILE @i <= @ub BEGIN ... SELECT @i = @i + 1 END</pre>

15.8.4 STORED PROCEDURE

To execute a stored procedure-

Oracle	Sybase ASE
<pre>Result := proc_name(param1 => NAME, param2 => value);</pre>	<pre>EXEC @Result = proc_name @param1 = @NAME, @param2 = @value output</pre>

15.8.5 DATE MANIPULATIONS

Oracle	Sybase ASE
<pre>TO_CHAR(start_date, 'FMHH12FM:MI PM')</pre>	<pre>SUBSTRING(RTRIM(CONVERT(CHAR, start_date, 0))), 13, 8)</pre>
<pre>TO_CHAR(sysdate, 'mm/ yy')</pre>	<pre>SUBSTRING(RTRIM(CONVERT(CHAR, GETDATE(), 3)), 4, 5)</pre>
<pre>SELECT add_months(xyz ,3) FROM dual</pre>	<pre>SELECT DATEADD(month, 3, xyz)</pre>
<pre>cmd := "credit_expiration_date = last_day(TO_DATE(' " + cc_date + "')) "</pre>	<pre>@cmd = "credit_expiration_date = CONVERT(DATETIME, RTRIM(CONVERT(CHAR, " + s_month + "))) + '/' + RTRIM(CONVERT(CHAR, DATEDIFF(DAY, RTRIM(CONVERT(CHAR, " + s_month + "))) +'/1/' + RTRIM(CONVERT(CHAR, " + nYear + "))),</pre>

Oracle	Sybase ASE
	<pre>DATEADD(MONTH, 1, RTRIM(CONVERT(CHAR," + s_month + ")) + '/1/' + RTRIM(CONVERT(CHAR," + nYear + ")))))) + '/' + RTRIM(CONVERT(CHAR," + nYear + "))) "</pre>

15.8.6 MISCELLANEOUS

Oracle	Sybase ASE
<pre>DECLARE COUNT INT; BEGIN SELECT COUNT() into COUNT FROM XYZ; END;</pre>	<pre>declare @COUNY int SELECT @COUNT = COUNT(*) FROM XYZ</pre>
<pre>SELECT NULL as start_date</pre>	<pre>SELECT CONVERT(DATETIME, NULL) as start_date SELECT '' as start_date (better)</pre>
<pre>SELECT NULL as course_id</pre>	<pre>SELECT CONVERT(INT, NULL) as course_id SELECT 0 as course_id (better)</pre>
<pre>SELECT NULL as first_name</pre>	<pre>SELECT CONVERT(CHAR, NULL) as first_name SELECT '' as first_name (better)</pre>

15.8.7 OUTER JOIN SYNTAX

Oracle	Sybase ASE
<pre>student.student_id = student_company.student_id (+)</pre>	<pre>student.student_id *= student_company.student_id</pre>
<pre>student_company.student_id (+) = student.student_id</pre>	<pre>student_company.student_id =* student.student_id</pre>

15.8.8 LIMITATIONS

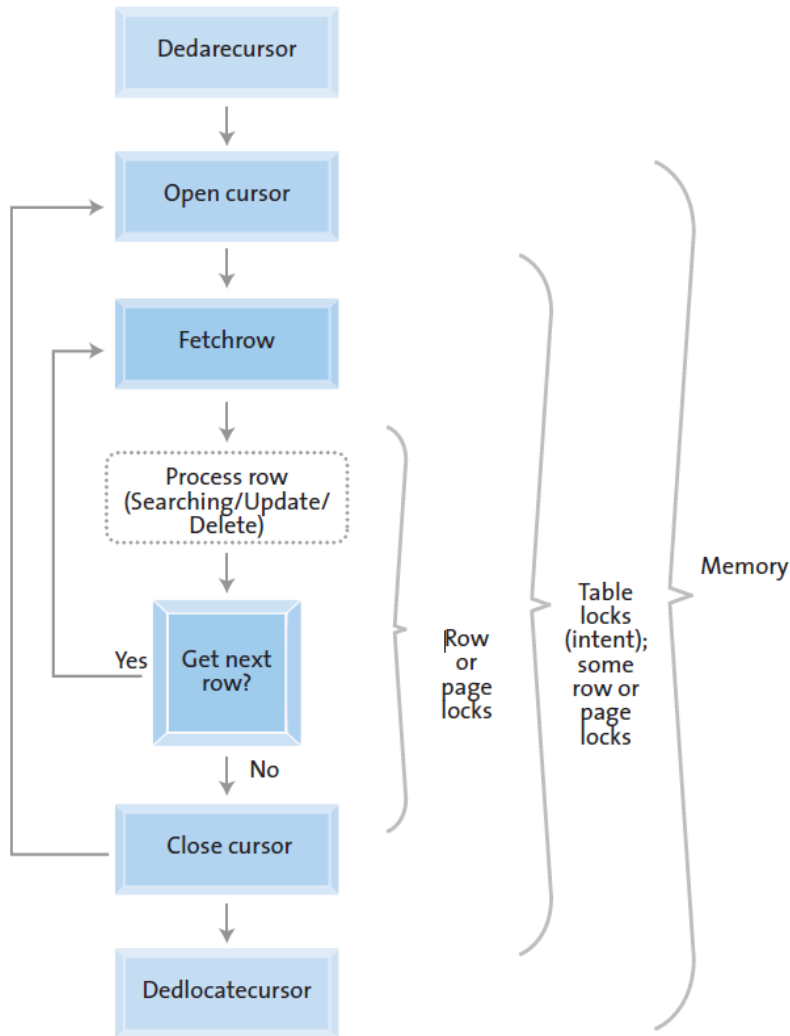
Sybase ASE does not allow another join relationship on a table that already has an outer join (Example #1). In addition, if you submit a query with an outer join and a qualification on a column from the inner table of the outer join, the results may be other than what you expect (Example #2). Ideally, the database design should be de-normalized to remove the need for these relationships.

Oracle	Sybase ASE
Example #1:	Example #1:
<pre>SELECT DISTINCT a.id, b.name, c.desc FROM a, b, c WHERE a.id = b.id (+) and b.id2 = c.id2 (+) (or b.id = c.id2) and a.code = 1 ORDER BY b.name</pre>	<pre>SELECT a.id, b.name, c.desc FROM a, b, c WHERE a.id = b.id and b.id2 *= c.id2 and a.code = 1 UNION SELECT a.id, '', '' FROM a WHERE a.code = 1 and (NOT EXISTS (SELECT 'X' FROM b WHERE a.id = b.id)) ORDER BY 2</pre>
Example #2:	Example #2:
<pre>SELECT a.id, b.name FROM a, b WHERE a.id = b.id (+) AND b.name LIKE 'Bill%'</pre>	<pre>SELECT a.id, b.name FROM a, b WHERE a.id *= b.id AND b.name LIKE 'Bill%'</pre>

16 APPENDIX E – ISSUES WITH 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 include 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, Sybase ASE 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 Sybase ASE 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 un-partitioned 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:

1. Optimize cursor selects using the cursor, not an ad hoc query:
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, Sybase ASE 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:

Sybase ASE 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.

17 APPENDIX F – 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 allpages-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 Sybase ASE 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 ASE 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 it 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 EXISTS to determine if the record in question exists, 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.

18 APPENDIX G – USING 3RD PARTY TOOLS

18.1 POWERDESIGNER

This is not exactly a 3rd party tool, but it is a tool external to Sybase ASE. PowerDesigner is arguably the most advanced data modeling tool in the market. With support for over 30 database types it is the most flexible data modeling tool on the market. This chapter will walk you through on how to install PowerDesigner to work with both Oracle and Sybase ASE.

The following has to be installed for using Power Designer:

- PowerDesigner 15.0 and higher
- Oracle ODBC driver

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 oracle 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 Oracle 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 ASE database name as name of file)

Before creating the schema script from a PDM, change the target DBMS to Sybase ASE, which converts the datatypes of the model.

Steps to change the target DBMS to Sybase ASE:

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

If the any of the table of the Oracle database uses the timestamp datatype, then change it to datetime datatype. For that right click on the table having timestamp datatype to select columns of the table.

Change the timestamp

datatype to datetime datatype.

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 D:

- For functions used in constraints like length in Oracle and char_length in Sybase ASE

If the number of fields in the tables exceeds the limit in Sybase ASE the tables will have to be split and all queries referencing the table may have to be modified as well.

If the size of the field exceeds the limit in Sybase ASE it will have to be split and all queries referencing the field may have to be modified.

Note that Power Designer converts BFILE datatype of oracle to IMAGE datatype of Sybase ASE. Change it to the required datatype if required.

Stored procedures and triggers have to be created manually using Appendix C –Datatype Differences and Appendix D - SQL Language Differences.

18.2 SWISSQL

This is one of the few tools that automatically migrate PL/SQL code into Sybase T-SQL code.

The tool supports the following code conversions:

- Procedures, Functions, Views and Packages from PL/SQL to Transact-SQL. Supports both Named and Anonymous PL/SQL blocks.
- Variables, Cursors defined in Package Specification and Package Body.
- Procedure calls made inside another procedure.
- PL/SQL built-in and user-defined Data types To Transact SQL.
- PL/SQL built-in Functions into corresponding Transact SQL Procedures.
- Built-in packages like DBMS_OUTPUT & DBMS_SQL.
- Cursor Declaration, OPEN, FETCH and CLOSE Statements, Cursor Variables and Implicit Cursors.
- REF Cursor variables into Transact-SQL.
- %ROWCOUNT, %FOUND, %NOTFOUND and %ISOPEN cursor attributes.
- Sequence Objects.
- PL/SQL Label.
- Scalar and Record Anchor variables.
- User Defined Exceptions.
- PL/SQL IN OUT parameters.
- Native Dynamic SQL Statements(NDS).
- Collection type.
- Transaction features - COMMIT, ROLLBACK and SAVEPOINT.
- Conditional and control statements.
- The tool provides mechanism to configure Data type mapping to be used during migration through a configuration file.

18.3 FACT

FACT is a high speed Oracle data export that allows parallel flat file creation. These files can be used as input for the Sybase ASE bulk copy tool bcp. A high-speed Oracle data export is needed in large database configuration. Using the Oracle spool method will not produce the desired outcome and in many cases the job might crash or hang before it can be completed. In addition to these problems, you have to write your own PL/SQL procedure to enable the data spooling.

18.4 ETL TOOLS

If you have an ETL tool already in use that supports both Oracle and Sybase ASE, you can use this tool to move data from Oracle to Sybase ASE. Please make sure that you adhere to any license restrictions and clear the use of this tool to move data from Oracle to Sybase ASE with this vendor.

19 APPENDIX H – USING BCP

Bulk copy utility (bcp) provided by is used to move data between Sybase ASE 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
[-f formatfile]
[-e errfile]
[-d discardfileprefix]
[-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]
[-W]
[-X]
[-M LabelName LabelValue]
[-labeled]
[-K keytab_file]
[-R remote_server_principal]
[-C]
[-V [security_options]]
[-Z security_mechanism]
[-Q]
[-Y]
[-y sybase_directory]
[-x trusted.txt_file]
[--maxconn maximum_connections]
[--show-fi]
[--hide-vcc]
[--colpasswd [[[database_name.[owner].table_name.]column_name
               [password]]]
[--keypasswd [[database_name.[owner].]key_name [password]]]
```

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 Sybase ASE login name.

-P password – specifies an Sybase ASE 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 Sybase ASE. If you do not specify -I, bcp looks for an interfaces file (sql.ini in Windows) located in the directory specified by the SYBASE ASE environment variable (ini directory in Windows).

-S server – specifies the name of the Sybase ASE 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 Sybase ASEs or between Sybase ASE 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 Sybase ASE 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 Sybase ASE.

Sybase ASE 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 Sybase ASE:

- 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”.

20 APPENDIX I – STEP BY STEP MIGRATION EXAMPLE

Throughout this guide you learned the differences between Oracle and Sybase ASE. The key to a successful migration is the knowledge of what to expect and meticulous planning. Failure to thoroughly analyze the source Oracle system will haunt you during the migration process.

Although this guide tries to explain many different aspects and differences between the two database systems, it is only a collection of the most common tasks, system comparisons and examples. There are many more Oracle configurations, PL/SQL constructs or SQL language exceptions that will make it difficult to perform a migration.

20.1 PLANNING THE MIGRATION

The first step of a migration is to thoroughly analyze the Oracle system that you want to migrate. You need to know everything about:

- Database configurations
- Database schema
- Physical data model implemented in the database (always reverse engineer from the production database)
- Extracting every stored procedure, function and package (always reverse engineer from the production database)
- Capture all the indexes physically present at the time of the analysis
- Capture all the connection points to client applications
- Analyze the SQL*Net configuration. Not everything translates 1:1 into Sybase's TDS protocol.
- Identify every Oracle component that does not have a migration path outlined in this guide. Look out for possible show stoppers and develop an alternative plan. If there's no obvious migration plan, maybe it is not needed in Sybase or you have to approach a single step task in Oracle with a multi-step task in Sybase.
- Analyze the data volume and develop a plan on how to migrate the data into the Sybase ASE server.
 - How many tables?
 - How many rows per table?
 - Which transfer mechanism is most appropriate for your environment? It could be a mix of many or all transfer mechanisms.

- How time sensitive is the date? Do you need the data up to the last commit represented in the Sybase ASE server or do you have some lag time?
- Outline every single step of the migration and who is responsible for.
- Plan to have the infrastructure of the Sybase ASE server completely up and running before starting any migration activity. You need to be 100% sure to cover all the database configuration, database backup, system configuration, security, schemas and maintenance tasks before transferring any data or granting access from client applications to allow developers to migrate their code. This approach ensures that issues are being detected at the correct level and not trickled from the client application into the database and at the end it is not clear what causes the issue, the database or the application.

20.2 INSTALLING THE SOFTWARE

20.2.1 SYSAM

Before you install the software you need to be aware of one major difference between Oracle and Sybase. Unlike Oracle, Sybase protects its software with a license key. This means that you need to have a license key available before you start the installation.

Sybase uses the SySAM software to manage the licenses. You have the choice between local and served licenses. The local license will be managed locally on the OS server that hosts the Sybase ASE server. In larger environments it is recommended to switch to a served license model. This means that all the licenses are managed on a central server and the Sybase ASE checks out its license from there. Sybase ASE checks out the license at boot time and then does a periodic heartbeat check in frequent intervals.

If you don't have a license, but need to get a jump start on the installation, you have a 30 day grace period where all the Sybase ASE options are available. After this grace period the Sybase ASE server fails to operate.

20.2.2 PRE-INSTALLATION

Before installing the software you need to be fully aware of the system requirements and the pre-installation requirements for Sybase ASE on your operating system. Sybase provides several manuals that you need to reference before you install the software.

Sybase offers an online repository for Sybase manuals at <http://infocenter.sybase.com>. You need to navigate to Sybase ASE server version you are planning to install. Once you opened the sub-tabs of the tree navigation, you need to find 2 distinct manuals.

- Release Bulletin Adaptive Server Enterprise <version> for <OS Type>
- Installation Guide Adaptive Server Enterprise <version> for <OS Type>

These 2 manuals contain all the pre-installation requirements you need to install Sybase ASE successfully on your target OS.

20.2.3 INSTALLATION

The same manuals will help you to install Sybase ASE from either a CD device or a compressed archive file that you downloaded from the Sybase download center. In either case, make sure that you completed the pre-installation tasks.

The installation can either be performed via a GUI guided process or via a scripted and automated background process. The latter is designed to install new Sybase ASE servers in large environments and guaranteeing company policies are met without a lot of manual intervention.

- The GUI installation will ask you to configure the following services and servers:
- Sybase ASE Server (the database server)
- Backup Server (to perform database and transaction backups and restores)
- Monitor Server (to capture performance metrics)
- XP Server (access to the OS command shell from within the database server)
- Job Scheduler (to schedule database jobs like backups, dbcc or index maintenance)
- Web Services (to allow web services API calls to the database server)
- Enhanced Full-Text Search (to enable full text search on the database server)

- Unified Agent (the communication agent that maintains an inter-process and inter-server communication for all Sybase products. This agent allows remote startup and shutdown of a Sybase ASE server)

The installation process copies all the files into the pre-defined folders below the root folder that is being set by the \$SYBASE environments variable. Sybase installs all its files below this root folder, including datafiles, if not specific set during the installation or if you're using raw devices.

The most crucial step in the installation is the buildmaster process. This process will create the master database and once created starts the Sybase ASE engine. This is the point when you need to be on the lookout for system failures. If there is an OS configuration that is not compatible with this Sybase ASE version, the Sybase ASE engine will fail to start and all subsequent installation steps will never happen.

20.3 MIGRATING THE USER LOGINS

Before you can use the Sybase ASE server you need to create user logins. But before creating new user logins you need to understand the login concept of Sybase ASE and the definition of the built in logins and users.

The master database contains a system table syslogins that contains all user logins for the entire Sybase ASE server. The login information does not contain object access permission, just the login credentials, password and other user specific information. Think about this as an authentication step. Once Sybase ASE validated the login, the controls are handed to the default database stored in the syslogins table.

At the database level the object access permissions are being attached to this login. This completes the login process and the user is able to work with the Sybase ASE objects.

There are special logins in Sybase ASE:

- **sa** – this is the super user access to Sybase ASE, comparable to the SYS account in Oracle. This user can access everything and some system activities require sa login.
- **dbo** – this is not a login, but a super user object access permission account. The sa login is automatically a dbo (database owner) on every database this Sybase ASE server controls. Any

user can have dbo rights on many different databases. There can be many dbo accounts on a single database. The dbo (database owner) account is comparable to the SYSTEM account.

- **guest** – this is the access permission if a Sybase ASE login does not have a user definition on a database and is not a dbo. The database must be configured to allow guest access. This is comparable to a PUBLIC role in Oracle.

20.4 MIGRATING THE SCHEMA

As mentioned before, reverse engineering of the schemas in the Oracle database is the only option that guarantees success. No matter how well your version control and change management is defined, there is always room for improvement. To capture all objects in every schema, extracting it directly from the database is the most reliable way.

You can use the DESC command to get the information on the database objects, but this requires a lot of coding and manual preparations. Extracting objects from the database is left best in the hand of tools. PowerDesigner is one of the tools. It allows the reverse engineering of all tables, indexes, triggers, functions and procedures. PowerDesigner is capable of converting datatypes into the matching counterpart when selecting a different target database than the source database. In this case the source database is Oracle and the target database is Sybase ASE.

If you opt for a low cost tool, TOAD is a good start to extract the objects. However, after extracting the objects, you need to convert the datatypes manually. If you have just a few tables, no big deal, but converting hundreds of tables can be cumbersome.

Please keep in mind that none of these tools will convert the PL/SQL language into T-SQL. There is a tool available that will do that for you. SwisSQL, this tool has a good conversion rate. Even with all the tools, you need to check the results for accuracy, and most of all, test, test, test.

Appendix G contains some 3rd party tools.

Another aspect of migrating the schema is the number of schemas you have to migrate. Each Oracle schema has to translate into a Sybase ASE database. This means that you have to create a login for the schema user and this login will become dbo (database owner) on the respective database. This way you can manage the permissions exactly as you have it in Oracle. If you use synonyms to access the tables you can translate these into views. If it is a public synonym then any login has access to the

underlying table. You could either place the view into the same database as the table or create a separate database that contains the views that then point to the table in the respective database.

Unfortunately there is no corresponding solution for synonyms all the other database objects that are not tables or views. In this case you need to either plan for code changes and call the objects with its full, qualified name or develop another workaround strategy.

Planning the correct schema migration approach is crucial to the success of the migration project. The goal is to migrate Oracle schemas to preserve the data and access security, meet performance goals and keep the impact on the source code to a minimum. This is a delicate balance and requires attention to details and some creativity.

20.5 MAPPING THE DATA TYPES

In most cases the mapping of data types is straight forward. It gets more difficult when special data types are being used or when database limits are being tested by very large data types. There is a direct impact of Sybase ASE's block size and the maximum size a specific data type can hold.

These Oracle data types exceed the maximum capacity of a corresponding Sybase ASE data type for the following data types:

- For any Sybase ASE block size
 - CLOB
 - BLOB
- Sybase ASE block size of 2k
 - CHAR
 - VARCHAR2
 - RAW
- Sybase ASE block size of 4k
 - VARCHAR2

With these limits in mind you can map your Oracle data types to the correct Sybase ASE data type. Unfortunately there are no data types available in Sybase ASE to cover the CLOB or BLOB data type. AN option would be an external proxy table that links filesystem files with database rows. This would be an option to overcome this limitation.

The tedious task of mapping all the Oracle data types in every table against the Sybase ASE data types can be mostly eliminated by using PowerDesigner to reverse engineer the Oracle database and then change the target database to Sybase ASE. All the 1:1 conversions happen automatically. Only the exceptions listed above will need some manual attention if the Oracle data type size is bigger than the smallest Sybase ASE data type size.

This is the foundation for any PL/SQL conversion and application migration. The same data type conversion rules apply to PL/SQL variables that need to be converted into T-SQL variables that adhere to the same rules as table column data types. There is also a 3rd party tool available that will mostly automate this step. SwisSQL is one of the leading PL/SQL to T-SQL conversion tools.

Appendix G lists some 3rd party tools.

20.6 UNLOADING AND LOADING THE DATA

Unfortunately there is no easy way to extract data from an Oracle database. Well, at least no built-in function is available. You have the choice between writing your own data extract PL/SQL procedure or use a 3rd party tool to unload the Oracle data.

Once you get over the challenge to extract the data from the Oracle database you need to pay close attention to some data type formats. Especially the data format differs from Oracle to Sybase ASE. You can either tell the extract process to format the date data types in a format Sybase ASE understands or you can try to find a Sybase ASE date conversion rule that will handle to date format in the export file.

If you're using either CIS with the Express Connect Data Access Option for Oracle or the Sybase Replication Server with Express Connect for Oracle, data format conversions are handled automatically with the tool. This is a huge benefit of these tools that is worth evaluating.

The move the data from an Oracle to a Sybase ASE system it is important to analyze both the total volume of data and the individual table sizes. The total data volume will determine how long the overall data move will take and the size individual table will determine the method of data movement.

Depending on how long it takes to load the data from Oracle to Sybase ASE you need to plan for multiple loads to capture the latest data changes from the Oracle database. The key is to identify the delta between the initial data load and subsequent incremental data loads. Very large databases probably need an initial data load that can take several days or weeks and then an additional delta load for the data that has been added or changed during the time of the initial load. These tasks can go through multiple iterations until the Sybase ASE server is in sync with the Oracle server.

You probably detected a twist in the example of loading data through multiple iterations. The focus is on changed data. Loading data fast requires an empty table and no logic to detect data that is already loaded. This only happens during the initial load, all subsequent delta loads require some sort of logic to detect already existing data and then switch from inserting data to updating data. This requires indexes in place to have a chance to complete this job in time, needless to say that this complicates things. I haven't even started to mention that changed data can mean deleted data. You can see that this will get out of hand quickly.

In any case it is not recommended to create scripts and processes for data movement purposes that will have to emulate an incremental data load. Not only do you have to write code that captures all the changed data since the initial data export. You have to write the code to determine if on the target Sybase ASE server the data is new data or changed data. Capturing deleted data from the source

Oracle database to the target Sybase ASE database is nearly impossible. You would have to compare the entire table between the 2 systems to determine any differences in data rows.

The following table will outline data move scenarios for specific data situations:

Scenario	Data Movement Method
<p>The recommended data movement method applies to the following scenarios:</p> <ul style="list-style-type: none"> • Small tables with less than 10 million rows. • The switch over window is long enough to completely refresh the entire database, even if there are a couple of large tables 	<p>For the initial load:</p> <ul style="list-style-type: none"> • CIS method <ul style="list-style-type: none"> • Copy the data from Oracle to Sybase ASE via CIS with an INSERT statement. • Flat File Method <ul style="list-style-type: none"> • Unload the data with a custom Oracle PL/SQL spooling procedure. • Load the data with the bcp tool. <p>For subsequent loads:</p> <ul style="list-style-type: none"> • CIS method <ul style="list-style-type: none"> • Delete the data from the target table. • Copy the data from Oracle to Sybase ASE via CIS with an INSERT statement. • Flat File Method <ul style="list-style-type: none"> • Unload the data with a custom Oracle PL/SQL spooling procedure. • Delete the data from the target table. • Load the data with the bcp tool. <p>This method allows synchronizing these tables between Oracle and Sybase ASE to be up-to-date in a very short period of time.</p>
<p>The recommended data movement method applies to the following scenarios:</p> <ul style="list-style-type: none"> • Large tables with more than 10 million rows. • The downtime window for the switch over is too small to move the entire database at one time. 	<p>Depending of the availability requirements of your systems, this data move might exceed the threshold to be completed entirely during the final switch over.</p> <p>It is recommended to enlist the help of tools to perform an incremental data move after the initial data load.</p> <p>For the initial load:</p> <ul style="list-style-type: none"> • Install Sybase Replication Server with Express Connect for Oracle • Establish a one –way replication for the selected tables • In order to minimize the impact on the servers the initial load of the data should be done outside the Replication Server. • CIS method <ul style="list-style-type: none"> • Copy the data from Oracle to Sybase ASE via CIS with an INSERT statement.

Scenario	Data Movement Method
	<ul style="list-style-type: none">• Flat File Method<ul style="list-style-type: none">• Unload the data with a custom Oracle PL/SQL spooling procedure.• Load the data with the bcp tool. <p>For incremental loads:</p> <ul style="list-style-type: none">• No further action necessary. The data in these tables are automatically in sync.

20.7 SETUP THE BACKUP

You need to analyze the backup methodologies used in the current Oracle database. This will determine the backup strategy for the Sybase ASE server. Oracle offers several backup methods that need to be translated into Sybase ASE backup strategies.

List of Oracle backup methods and their Sybase ASE counter parts:

- **Import / Export**
 - This backup is equivalent with exporting the data from a Sybase ASE server with bcp and managing the DDL, rules, types and defaults with the defncopy command.. The imp and exp commands can import/export the entire database, individual schemas or a single table.
 - To export the data from a Sybase ASE server with bcp, you need to create a script that emulates the Oracle exp command to allow for multi database bcp out and single database bcp out. Individual tables are the default mode for bcp. The bcp command has to be accompanied by the defncopy command.
- **Data Pump**
 - This is the new import / export feature since Oracle 10g and enables high speed import and export. The basic functionality is identical with the old imp and exp commands.
 - The same bcp methods for data exports have to be applied.
- **RMAN**
 - The Oracle Recovery Manager (RMAN), a command-line and Enterprise Manager-based too, is the Oracle-preferred method of efficiently backing up and recovery an Oracle database. It packages al the pre-backup preparation task, the core backup functions and the backup cleanup function in one tool. It removes dependencies on OS and SQL*Plus scripts.

When used in the Oracle Enterprise Manager (OEM), RMAN can be setup and scheduled to backup the Oracle database. The setup parameters are very flexible and allow for several levels of cleanup steps to keep the backup overhead at a minimum.

- Sybase ASE offers the dump command for backups and the load command for restores. Unlike Oracle, Sybase ASE does not have an all-in-one tool to perform the backup tasks. You have to embed the dump command into a script to perform the pre-backup and the cleanup tasks and you need to invoke the Job Scheduler to schedule backup scripts.

However, Sybase ASE offers greater flexibility with the backup server. In order for the dump command to work the Sybase Backup Server needs to be up and running. The Sybase Backup Server is an external process to the Sybase ASE server and maintains its own security. You have to start the Sybase Backup Server separately after Sybase ASE starts. Within Sybase ASE the Sybase Backup Server is defined as a remote server.

The dump command executes the block backup on a single database and transfers the data to the Sybase Backup Server. The Backup Server then compresses the data, if requested, and writes the data either to tape or a file on a disk drive. Each database has to be backed up separately. During the backup process the Sybase ASE database is fully accessible and writable. The dump command ensures data consistency and integrity during the duration of the dump command. When restoring the data via the load command, all completed transactions that occurred during the dump command will be applied with the restore.

Because the Sybase Backup Server is treated as a remote server, you can plan for centralized backup servers that collect backup data from dump commands over the network. This eliminates the need for mounting backup filesystems.

- **Archive Log Mode**

- In Oracle you have to make a choice when setting up the backup system. Because the redo log files are managed in a round robin fashion and are being reused, redo log files will get overwritten. Oracle supports cold backups; the database must be shutdown to perform this action, and the hot backup method. Hot backups allow users to continue using the database while backups are being executed.

The definitions below will explain the differences between the two archive log modes:

- **Running an Oracle Database in NOARCHIVELOG Mode**

- When you run your database in NOARCHIVELOG mode, you disable the archiving of the redo log. The database control file indicates that filled groups are not required to be archived. Therefore, when a filled group becomes inactive after a log switch, the group is available for reuse by LGWR.
 - NOARCHIVELOG mode protects a database from instance failure but not from media failure. Only the most recent changes made to the database, which are stored in the online redo log groups, are available for instance recovery. If a media failure occurs while the database is in NOARCHIVELOG mode, you can only restore the database to the point of the most recent full database backup. You cannot recover transactions subsequent to that backup.
 - In NOARCHIVELOG mode you cannot perform online tablespace backups, nor can you use online tablespace backups taken earlier while the database was in ARCHIVELOG mode. To restore a database operating in NOARCHIVELOG mode, you can use only whole database backups taken while the database is closed. Therefore, if you decide to operate a database in NOARCHIVELOG mode, take whole database backups at regular, frequent intervals.
- **Running an Oracle Database in ARCHIVELOG Mode**
 - When you run a database in ARCHIVELOG mode, you enable the archiving of the redo log. The database control file indicates that a group of filled redo log files cannot be reused by LGWR until the group is archived. A filled group becomes available for archiving immediately after a redo log switch occurs.
 - The archiving of filled groups has these advantages:
 - A database backup, together with online and archived redo log files, guarantees that you can recover all committed transactions in the event of an operating system or disk failure.
 - If you keep an archived log, you can use a backup taken while the database is open and in normal system use.
 - You can keep a standby database current with its original database by continuously applying the original archived redo logs to the standby.
 - You can configure an instance to archive filled redo log files automatically, or you can archive manually. For convenience and efficiency, automatic archiving is usually best.
 - If all databases in a distributed database operate in ARCHIVELOG mode, you can perform coordinated distributed database recovery. However, if any database in a distributed database is in NOARCHIVELOG mode, recovery of a

global distributed database (to make all databases consistent) is limited by the last full backup of any database operating in NOARCHIVELOG mode.

○ **Differences concerning backups**

Oracle No archive log

Must backup entire database

DB must be shut down

Only entire DB can be restored

In case of a failure, all changes since the last backup will be lost

Oracle Archive log

Can backup parts of database (datafiles tablespaces)

hot backups possible

Tablespaces can be restored

All committed transactions will be restorable

○ **Sybase ASE**

- Sybase ASE always performs a hot backup. No need to configure anything. This is the same functionality as Oracle's Archive Log, but no archive file cleanup is necessary.
- Because the transaction logs are part of the database, you need to perform periodical transaction log backups to enable up to the last commit recovery. Oracle keeps transaction logs in the online redo log files and the archive logs, you need to protect these files separately from media failure. Sybase ASE allows database internal mirroring of transaction logs to protect against media failure and enabling up to the last commit recovery capability.

● **Flashback**

- Oracle Flashback Technology is a group of Oracle Database features that let you view past states of database objects or to return database objects to a previous state without using point-in-time media recovery.
- With flashback features, you can do the following:
 - Perform queries that return past data
 - Perform queries that return metadata that shows a detailed history of changes to the database
 - Recover tables or rows to a previous point in time
 - Automatically track and archive transactional data changes

- Roll back a transaction and its dependent transactions while the database remains online
- Oracle makes flashback information available to applications through a flashback query option.
- Sybase ASE offer similar features with a combination of tools. Unlike Oracle, these tools are reserved for the system administrator and do not allow for application user retrieval.

To enable flashback like behavior in Sybase ASE you need to install and enable the following tools and features:

- **Auditing**

- The Sybase ASE auditing system allows you to log events on a system level down to the individual user level. You can establish auditing for events such as:
 - Server-wide, security-relevant events
 - Creating, deleting, and modifying database objects
 - All actions by a particular user or all actions by users with a particular role
 - Granting or revoking database access
 - Importing or exporting data
 - Log ins and log outs
- This will provide you with the necessary information to establish a timeline of events and gives powerful insights into 'what happened?' situations.

- **Archive Database**

- Archive database access allows a database administrator to validate or selectively recover data from a database dump (an "archive") by making the dump appear as if it were a traditional read-only database; this type of database is called an "archive database."
- Unlike a traditional database, an archive database uses the actual database dump as its main disk storage device, with a minimum

amount of traditional storage to represent new or modified pages that result from the recovery of the database dump. A database dump already contains the images of many (or even most) of the database pages, therefore, an archive database can be loaded without having to use Backup Server to transfer pages from the archive to traditional database storage. Consequently, the load is significantly faster than a traditional database.

- Without the archive database the only option you have to access past data is to create a new database with the same exact size as the database you want to restore and then loading the database dump files into the this database. This process can take a significant period of time to complete. A much faster and more effective way is to use an archive database.
 - With the timeline information from the auditing you will be able to determine exactly at what time the database integrity has been compromised by a disastrous drop table command, faulty transactions or other data manipulating events.
 - Now you can create the archive database with the database dump files and then applying the individual transaction dumps to the exact point in time with the `until_time` parameter of the `load transaction` command. Of course the restore time should be slightly before the time of the error.
 - The only flashback feature not covered by Sybase ASE is the rollback transaction feature. Although all the necessary information are available; audit information to determine which transaction you want to rollback and the archive database with the transaction log to determine which transactions and what data. Oracle uses a function called `LOG_MINER` to perform this action. This shouldn't be too difficult for Sybase to offer this functionality in a future Sybase ASE release.
- Needless to say that both, Oracle's flashback and Sybase's auditing, have to be enabled **before** a problem happens. For both database servers there are significant performance considerations to follow and you need to weight the benefits against the drawbacks.

As outlined in the previous paragraphs there are many backup options available in Sybase ASE to create a backup & restore system that will meet your requirements. No matter which method you prefer, you need to implement at least one to protect yourself from costly failures.

20.8 SETUP MAINTENANCE TASKS

Maintenance tasks are determined by database architectures and very specific to the application requirements. There are so many external factors that influence the type of maintenance as well as how many maintenance tasks are involved for any given system; we limited the topic to two of the most likely maintenance tasks. We already discussed the backup and recovery in the previous chapter.

- **Index Maintenance**

Maintenance on indexes can happen on several levels depending on how much data change a database experiences. The more new data gets entered into the system the more likely the statistics will be outdated to represent the correct cardinality in an indexed column, whereas updates and deletes tend to defragment indexes.

The following index maintenance task should be performed:

- **Index Rebuilds**

Rebuilding a clustered index on an allpages lock table will defragment the table and the index by re-ordering the table content by the column(s) in the clustered index. A clustered index on an allpages lock table is the data.

Rebuilding any other indexes, including a clustered index on datapages or datarows lock tables, will defragment the index only. The underlying data in the table remains untouched.

If you have very large tables with hundreds of millions of rows. It is recommended that you are considering using Sybase's partition option to reduce the time such index rebuilds will take. Without the partition option Sybase ASE will issue a full table lock on the table that the index is being created. With Sybase's semantic partitions you will be able to re-create local indexes on a single partition and schedule multiple index jobs in parallel. This will ease the locking of a table while an index build is in progress.

Scheduling regular index rebuilds on a system with moderate to large data growth is good practice and should be part of your maintenance task routine. You should schedule index rebuilds at times when the system load is light and no users are online. Specifically when you rebuild very large non-partitioned tables that will trigger a full

table lock and will block any other access.

- **Update Statistics**

The update statistics command updates column-related statistics such as histograms and densities. Statistics must be updated on those columns where the distribution of keys in the index changes in ways that affect the use of indexes for your queries.

Running update statistics requires system resources. Like other maintenance tasks, you should schedule it during at times when the load on the server is light. In particular, update statistics requires table scans or leaf-level scans of indexes, may increase I/O contention, may use the CPU to perform sorts, and uses the data and procedure caches. Use of these resources can adversely affect queries running on the server.

Using the sampling feature can reduce resource requirements and allow more flexibility when running this task.

In addition, some update statistics commands require shared locks, which may block updates.

Update statistics is your first line of defense in a performance degradation situation. Outdated statistics can mislead the query optimizer and produce wrong query plans. With the update statistics command you will be able to provide the best information to the query optimizer. You can run update statistics on all columns in each index of a table or a subset of it.

The update statistics syntax below illustrates the possibilities:

```
update statistics table_name
[[ partition data_partition_name ] [ (column_list ) ] |
index_name [ partition index_partition_name ] ]
[ using step values ]
[ with consumers = consumers] [, sampling=percent]
```

```
update index statistics
table_name [[ partition data_partition_name ] |
```

```
[ index_name [ partition index_partition_name ] ] ]  
[ using step values ]  
[ with consumers = consumers ] [, sampling=percent]  
    update all statistics table_name  
[ partition data_partition_name ]  
[ sp_configure histogram tuning factor, <value>  
    update table statistics  
table_name [partition data_partition_name ]  
    delete [ shared ] statistics table_name  
[ partition data_partition_name ]  
[( column_name[, column_name ] ...)]
```

For update statistics:

- **table_name** – generates statistics for the leading column in each index on the table.
- **table_name index_name** – generates statistics for all columns of the index.
- **partition_name** – generates statistics for only this partition.
- **partition_name table_name (column_name)** – generates statistics for this column of this table on this partition.
- **table_name (column_name)** – generates statistics for only this column.
- **table_name (column_name, column_name...)** – generates a histogram for the leading column in the set, and multicolumn density values for the prefix subsets.
- **using step values** – identifies the number of steps used. The default is 20 steps. To change the default number of steps, use sp_configure.
- **sampling = percent** – the numeric value of the sampling percentage, such as 05 for 5%, 10 for 10%, and so on. The sampling integer is between zero (0) and one hundred (100).

For update index statistics:

- **table_name** – generates statistics for all columns in all indexes on the table.
- **partition_name table_name** – generates statistics for all columns in all indexes for the table on this partition.
- **table_name index_name** – generates statistics for all columns in this index.

For **update all statistics**:

- **table_name** – generates statistics for all columns of a table.
- **table_name partition_name** – generates statistics for all columns of a table on a partition.
- **using step values** – identifies the number of steps used. The default is 20 steps. To change the default number of steps, use `sp_configure`.

The `datachange` function measures the amount of change in the data distribution since the last update statistics command was executed. Instead of blindly scheduling update statistics on every table at a given interval, you can selectively choose which index is in need of an update statistics refresh. Data changes are represented in a percentage number of the number of rows in a table. Because `datachange` is a T-SQL function, it allows you to script update statistic runs based on a certain data change percentage threshold.

This is an example of using `datachange` to trigger an update statistics:

```
select  @datachange = datachange("authors","author_ptn2", "city")
if @datachange >= 50
begin
    update statistics authors partition author_ptn2(city)
end
go
```

Update statistics can be scheduled the job scheduler and many threshold parameters are available to manage the update statistics needs combined with an on demand execution based on the `datachange` function.

- **DBCC (Database Consistency Checker)**

One of the most used maintenance task commands is probably DBCC. It checks the logical and physical consistency of a database and provides statistics, planning and repair functionality.

What was once a single threaded, monolithic process that had the potential to block access to the entire database for hours, sometimes days, is now a high performance, multi-threaded, parallel execution that can execute complex commands in a matter of minutes on very large databases.

One of the first steps you need to do to enable high-speed DBCC. You need to create the dbccdb. The dbccdb records configuration information, operation activity and the results of the DBCC operations. DBCC can use large I/O and asynchronous prefetch if you configure these options for the caches used by the databases or the objects to be checked. Using large I/O can dramatically increase the DBCC performance. As well as allocation worker process resources to increase the parallelism for DBCC. Be careful, massive parallel DBCC execution has the potential to “max out” the system resources, which has an impact on the overall performance.

DBCC performs the following checks:

Checks performed	checkstorage	checktable	checkdb	checkalloc	indexalloc	tablealloc	checkcatalog	textalloc
Allocation of text valued columns	X							X
Index consistency		X	X					
Index sort order		X	X					
OAM page entries	X	X	X	X	X	X		X
Page allocation	X			X	X	X		X
Page consistency	X	X	X					
Pointer consistency	X	X	X					
System tables							X	X

Text column chains	X	X	X					X
Text valued columns	X	X	X					X

DBCC is the most powerful tool for a DBA when facing a broken system. In some cases it is the only tool that will provide repair functionality. Every error is different in nature and different DBCC repair strategies and commands are needed. Please refer to the Sybase ASE product manuals and the system administrator guide for more details on DBCC.

20.9 SETUP MONITORING

If there is a single functionality that is a 3rd party vendor favor, then it would be database monitoring. Both Oracle and Sybase ASE can point to a long list of products that offer database monitoring. In this case we will take a look at monitoring options provided natively by the database systems.

20.9.1 ORACLE

Oracle Enterprise Manager (OEM) or also called grid control contains an extensive list of database monitoring functionalities out of the box and ready to use. Of course there is always the script based option to gather vital system monitoring information.

The two main monitoring functions are system alerts and performance monitoring. For alerting Oracle uses a rules based system that defines thresholds, applies policies that can trigger corrective measures or triggering a launch of a job, plus notification of recipients via email.

Typical Alerts Groups are:

- Agent Upload Problems
 - System-generated notification rule for monitoring Agents who may have problems uploading data to the Management Service.
- Agents Unreachable
 - System-generated notification rule for monitoring Agents who lose contact with the Management Service due to network problems, host problems or Agents going down.
- Database Availability and Critical States
 - System-generated notification rule for monitoring Databases' availability and critical metric statuses.
- Host Availability and Critical States
 - System-generated notification rule for monitoring Hosts' availability and critical metric statuses.
- Listener Availability
 - System-generated notification rule for monitoring database Listeners' availability and critical metric statuses.
- PAF Status Notification
 - System-generated notification rule for Provisioning Advisor Framework: Notifies the instance creator of any status updates.

- Provisioning Engine Notification
 - System-generated notification rule for provisioning engine: Notifies the provisioning engine about any change in job status submitted by it.

The performance monitoring is managed by Oracle's Automatic Database Diagnostic Monitor (ADDM). It will use statistics collected by the automatic workload repository (AWR) to rank the problems and the recommendations according to the DB time statistics.

ADDM will report on the following

- Expensive SQL/Java statements
- I/O performance issues
- Locking and Concurrency issues
- Excessive parsing
- High checkpoint load
- Resource bottlenecks, including memory and CPU bottlenecks
- Undersized memory allocations
- Connection management issues, such as excessive logon/logoff activity

The report itself will contain

- Expert problem diagnosis
- Emphasis on the root cause of the problem rather than on the symptoms
- A ranking of the effects of the problems, which means you can quickly find the problem
- Recommendations ranked according to their benefit

20.9.2 SYBASE ASE

Sybase uses a combination of tools to provide alerting and performance monitoring functionalities. Unlike Oracle where all the monitoring functionalities are located in OEM, Sybase took the independent tool approach. This allows for more flexibility, but complicates the use of the tools by having to navigate several tools GUI.

20.9.2.1 SYBASE MONITOR SERVER

The Sybase Monitor Server allows capturing information from the server configuration, the database design information and the SQL statements and stored procedures executed on this server.

The monitoring happens in real time and you can view the results through the Monitor Viewer in Sybase Central. The Monitor Server also provides a client API that allows for custom development of monitoring applications.

The Sybase Monitor Server must be configured and started using its own run file and parameters. This is similar to the backup server configuration.

20.9.2.2 SYBASE HISTORICAL SERVER

In order to view historical performance information the Sybase Monitor Server enlists the services for the Sybase Historical Server. The historical performance information will be stored for deferred analysis. The data is stored in files.

Sybase Historical Server also has an added feature of recording sessions and transactions. This allows for playback at a later time or in a different server. Besides having a great tool to capture activities on a server, this allows to capture trouble sessions on production and replay it on a test server.

20.9.2.3 SYBASE CONTROL CENTER

This is a web based performance monitoring and management tool. Sybase Control Center maintains its own repository and is capable of managing up to 50 servers. The Sybase Control Center chapter in this guide contains all the details.

20.10 MIGRATING TRIGGERS AND STORED PROCEDURES

Migrating PL/SQL to T-SQL is probably the most involved part of any Oracle to Sybase migration. You will not only dealing with data type migrations for variables; you will be facing ANSI SQL transaction processing issues, converting global variables in packages into either a temp table approach or a permanent table with session awareness. No matter how you slice it, there is a lot of work ahead of you.

The main difference between Oracle and Sybase is the transaction handling. Oracle is ANSI SQL based and Sybase use T-SQL as its standard. Oracle's transactions are implicit whereas Sybase handles transactions explicit. This means that Oracle automatically opens a transaction whenever an INSERT, UPDATE or DELETE statement appears. In Sybase you have to explicitly issue a BEGIN TRAN command to start a transaction. The details of this behavior have been documented throughout this guide.

The key for a successful stored procedure migration is to identify and separate packages into individual stored procedures while maintaining the integrity of the global variables that are being used. If the package is using REFCURSOR variables, more work has to be done to translate this behavior into Sybase. Although Sybase does support cursors in its SQL language, the REFCURSOR concept is not known to Sybase and an alternative approach has to be developed.

Translating REFCURSOR constructs into T-SQL views and cursor processing is the most straight forward approach as outlined in the PL/SQL and Trigger SQL Language Construct chapter.

Within triggers the biggest difference is that ORACLE supports BEFORE STATEMENT and AFTER STATEMENT triggers as well as BEFORE ROW and AFTER ROW Triggers. Sybase only supports AFTER statement triggers. However, with the pseudo tables inserted and deleted, Sybase is able to identify before and after changes and is able to compare calculated columns, Oracle needs to implement BEFORE STATEMENT triggers in combination with AFTER ROW triggers to achieve similar results.

As stated before it is absolutely necessary to extract the stored procedures and triggers directly from the production database to guarantee that the code is exactly what the production system is running on. This will eliminate any surprises or ghost hunts when the migration is completed, but the results are not the same.

Please Appendix D for all the language differences and most importantly, you need a good search and replace tool that can execute changes across many files.

20.10.1 SQL HINTS

One important difference between Oracle SQL and Sybase ASE SQL is the extensive hint feature in Oracle. SQL statements in ORACLE use a special form of a comment to place hints for the query optimizer. Essentially overwriting the method on how the query optimizer compiles a query plan. This is a widely used technique in Oracle that needs special attention during the migration process.

Hints can be identified by a special comment right after the SELECT keyword in a SQL statement.

For example:

```
SQL> explain plan for
      2 select /*+ index (a) */
      3 object_name
      4 from my_objects a
      5 where object_type = 'INDEXTYPE';
```

Sybase ASE does not support Oracle style hints and all HINT references must be removed.

Use this list of Oracle Hints to find the keywords in SQL statments:

Optimization Goals and Approaches (2)	Access Path Hints (17)	Other (20)	Join Operation (7)
ALL_ROWS FIRST_ROWS RULE	CLUSTER FULL HASH INDEX NO_INDEX INDEX_ASC INDEX_DESC INDEX_COMBINE INDEX_JOIN INDEX_FFS INDEX_SS INDEX_SS_ASC INDEX_SS_DESC	APPEND NOAPPEND CACHE NOCACHE CURSOR_SHARING_EXACT DRIVING_SITE DYNAMIC_SAMPLING MODEL_MIN_ANALYSIS MONITOR NO_MONITOR OPT_PARAM PUSH_PRED NO_PUSH_PRED	USE_HASH NO_USE_HASH USE_MERGE NO_USE_MERGE USE_NL USE_NL_WITH_INDEX NO_USE_NL

	NATIVE_FULL_OUTER_JOIN NO_NATIVE_FULL_OUTER_JOIN NO_INDEX_FFS NO_INDEX_SS	PUSH_SUBQ NO_PUSH_SUBQ PX_JOIN_FILTER NO_PX_JOIN_FILTER QB_NAME RESULT_CACHE NO_RESULT_CACHE	
--	--	--	--

Join Order (2)	Query Transformation (13)	XML (2)	Parallel Execution (5)
ORDERED LEADING	FACT NO_FACT MERGE NO_MERGE NO_EXPAND USE_CONCAT REWRITE NO_REWRITE NOREWRITE UNNEST NO_UNNEST STAR_TRANSFORMATION NO_STAR_TRANSFORMATION NO_QUERY_TRANSFORMATION	NO_XMLINDEX_REWRITE NO_XML_QUERY_REWRITE	PARALLEL NOPARALLEL NO_PARALLEL PARALLEL_INDEX NO_PARALLEL_INDEX NOPARALLEL_INDEX PQ_DISTRIBUTE



Sybase supports its own form of hints. Unlike Oracle which has an array of options in the hint commands, Sybase allows you to force the use of an index and the degree of parallelism, as well as the join order in a complex query. Oracle combines all hints in a /*+ hint +/- format where you can simply line up the hint commands. Sybase uses a combination of in-line command extensions and session level SET commands.

Parallelism in Sybase is typically handled by server wide configurations and session level overwrites. Sometimes you need to get more granular than that and that's when the parallelism hint comes into play.

20.10.1.1 PARALLELISM HINT EXAMPLE IN SYBASE ASE

```
select...
    [from {tablename}
        [(index index_name
            [parallel [degree_of_parallelism | 1]]
            [prefetch size] [lru|mru]]],
        {tablename} [(index_name
            [parallel [degree_of_parallelism | 1]]
            [prefetch size] [lru|mru]]] ...
```

To specify a:

Parallel partition scan
Parallel index scan
Serial table scan
Serial index scan
Parallel scan, with the choice of table or index scan left to the optimizer
Serial scan, with the choice of table or index scan left to the optimizer

Use:

(index tablename parallel N)
(index index_name parallel N)
(index tablename parallel 1)
(index index_name parallel 1)
(parallel N)
(parallel 1)

To specify a specific index to be used in a query, you need to add the (index) parameter to the SQL statement. In Sybase you can use the (index *index_name*) extension in select, update and delete statements.

```
select select_list
    from table_name [correlation_name]
        (index {index_name | table_name } )
        [, table_name ...]
    where ...
```

```
delete table_name
    from table_name [correlation_name]
        (index {index_name | table_name }) ...
```

```
update table_name set col_name = value
    from table_name [correlation_name]
        (index {index_name | table_name })...
```

For example:

```
select pub_name, title
  from publishers p, titles t (index date_type)
 where p.pub_id = t.pub_id
    and type = "business"
    and pubdate > "1/1/93"
```

20.10.2 JOIN ORDERS

To influence join orders you need to use the session level SET command

```
set forceplan [on|off]
```

This will lock in the join order as stated in the SQL statement. It will not change the join type. You can use the set forceplan command in combination with the forcing the index.

Please keep in mind that forcing changes in the query optimizer behavior is a temporary approach and if you upgrade to another version of Sybase, you may encounter a change in the query optimizer and the forced hints will actually work against you.

20.11 MIGRATING APPLICATION CONNECTIONS THROUGH JDBC

Migrating JDBC connections from Oracle to Sybase requires understanding how Oracle manages JDBC drivers vs. Sybase ASE. This will determine your approach on how to migrate JDBC.

Oracle provides the following JDBC drivers:

- Thin driver

It is a pure Java driver used on the client-side, without an Oracle client installation. It can be used with both applets and applications.

- Oracle Call Interface (OCI) driver

It is used on the client-side with an Oracle client installation. It can be used only with applications.

- Server-side Thin driver

It is functionally similar to the client-side Thin driver. However, it is used for code that runs on the database server and needs to access another session either on the same server or on a remote server on any tier.

- Server-side internal driver

It is used for code that runs on the database server and accesses the same session. That is, the code runs and accesses data from a single Oracle session.

Oracle recommends the use of its Thin JDBC driver in all and any cases when connections are made through TCP/IP. Focusing on this type of JDBC connection, Oracle offers a variety of connection options.

The JDBC Thin client is a pure Java, Type IV driver. It is lightweight and easy to install. It provides high performance, comparable to the performance provided by the JDBC Oracle Call Interface (OCI) driver. The JDBC Thin driver is written entirely in Java, and therefore, it is platform-independent. Also, this driver does not require any additional Oracle software on the client-side.

The JDBC Thin driver communicates with the server using TTC, a protocol developed by Oracle to access data from Oracle Database. It can be used for application servers as well as for applets. The driver allows a direct connection to the database by providing an implementation of TCP/IP that

implements Oracle Net and TTC on top of Java sockets. Both of these protocols are lightweight implementation versions of their counterparts on the server. The Oracle Net protocol runs over TCP/IP only.

The JDBC Thin driver can be used on both the client-side and the server-side. On the client-side, drivers can be used in Java applications or Java applets that run either on the client or in the middle tier of a three-tier configuration. On the server-side, this driver is used to access a remote Oracle Database instance or another session on the same database.

20.11.1 ORACLE JDBC EXTENDED DATA SOURCES

These extended data sources are not supported by Sybase ASE.

Name	Type	Description
connectionCacheName	String	Specifies the name of the cache. This cannot be changed after the cache has been created.
connectionCacheProperties	java.util.Properties	Specifies properties for implicit connection cache.
connectionCachingEnabled	Boolean	Specifies whether implicit connection cache is in use.
connectionProperties	java.util.Properties	Specifies the connection properties.
driverType	String	Specifies Oracle JDBC driver type. It can be one of oci, thin, or kprb.
fastConnectionFailoverEnabled	Boolean	Specifies whether Fast Connection Failover is in use.
implicitCachingEnabled	Boolean	Specifies whether the implicit statement connection cache is enabled.
loginTimeout	int	Specifies the maximum time in seconds that this data source will wait while attempting to connect to a database.
logWriter	java.io.PrintWriter	Specifies the log writer for this data source.

Name	Type	Description
maxStatements	int	Specifies the maximum number of statements in the application cache.
serviceName	String	Specifies the database service name for this data source. Specifies the TNS entry name, relevant only for the OCI driver. The TNS entry name corresponds to the TNS entry specified in the tnsnames.ora configuration file. This property is only for OracleXADatasource.
tnsEntry	String	Enable this OracleXADataSource property when using the Native XA feature with the OCI driver, to access Oracle pre-8.1.6 databases and later. If the tnsEntry property is not set when using the Native XA feature, then a SQLException with error code ORA-17207 is thrown Specifies the URL of the database connection string. Provided as a convenience, it can help you migrate from an older Oracle Database. You can use this property in place of the Oracle tnsEntry and driverType properties and the standard portNumber, networkProtocol, serverName, and databaseName properties.
url	String	Allows an OracleXADataSource using the Native XA feature with the OCI driver, to access Oracle pre-8.1.6 databases and later. If the nativeXA property is enabled, be sure to set the tnsEntry property as well. This property is only for OracleXADatasource.
nativeXA	Boolean	This DataSource property defaults to false.
ONSConfiguration	String	Specifies the ONS configuration string that is used to remotely subscribe to FaN/ONS events.

20.11.2 SYBASE ASE JDBC

You can use JDBC with the Adaptive Server SQL interface in either of two ways:

- **JDBC on the client** – Java client applications can make JDBC calls to Sybase ASE using the Sybase jConnect JDBC driver.
- **JDBC on the server** – Java classes installed in the database can make JDBC calls to the database using the JDBC driver native to Sybase ASE.

The use of JDBC calls to perform SQL operations is essentially the same in both contexts.

To enable Sybase JDBC connections across servers, the same principles have to be established as in Oracle. The TCP/IP connections are valid from the client to the web server and then the web server needs to connect to the database server via TDS.

20.11.3 JDBC STANDARD DATA SOURCES

These standard data sources are common in both Oracle and Sybase ASE.

Name	Type	Description
databaseName	String	Name of the particular database on the server. Also known as the SID in Oracle terminology.
dataSourceName	String	Name of the underlying data source class. For connection pooling, this is an underlying pooled connection data source class. For distributed transactions, this is an underlying XA data source class.
description	String	Description of the data source.
networkProtocol	String	Network protocol for communicating with the server. For Oracle, this applies only to the JDBC Oracle Call Interface (OCI) drivers and defaults to tcp.
password	String	Password for the connecting user.
portNumber	int	Number of the port where the server listens for requests
serverName	String	Name of the database server
user	String	Name for the login

20.12 MIGRATING SQL CODE INSIDE THE APPLICATION

The most difficult part of migrating SQL code inside an application is to identify all occurrences of SQL code. If the code is in a version control system, you can search for keywords like SELECT, INSERT, UPDATE and DELETE. Some applications pool the SQL code to isolate the database code from the business logic.

Both Oracle and Sybase ASE support native API calls in Java and so called embedded SQL for languages like C, C++ and COBOL.

21 INDEX

\$DSQUERY, 94, 96
\$ORACLE_HOME, 94, 95, 96
\$ORACLE_SID, 94, 97
\$SYBASE, 94, 95, 96, 97, 268

%ROWTYPE, 229

@@error, 170, 171, 181, 239
@@rowcount, 170, 171, 182, 237, 239
@@sqlstatus, 237
@@version, 71

ADDM, 290
AFTER, 31, 146
AFTER ROW, 212, 292
AFTER STATEMENT, 292
alarms, 196, 197
Alerts, 208, 289
ALL_TAB_COLUMNS, 70
ALL_TABLES, 70
Allpages, 53, 185
alter session, 74
alter table, 61, 74, 77, 193
ANSI transaction, 214
API, 25, 79, 132, 192, 193, 204
Application Components, 12
Architecture, 16, 99, 107
Archive Database, 280
Archive Log, 277, 279
ARCHIVELOG, 278
ASSIGNMENT, 171
asynchronous prefetch, 287
Auditing, 280
authentication, 57, 192
automatic memory management, 22

B*Tree, 65, 66

Background processes, 23
Backup, 25, 63, 88, 89, 100, 101, 102, 103, 192
Backup Server, 89, 100, 101, 102, 192, 267, 277, 281
bcp, 77, 79, 80, 82, 83, 95, 106, 114, 115, 197, 257, 259,
260, 261, 262, 263, 264, 274, 275, 276
BCP, 13, 115, 259
BEFORE, 31, 146, 232
BEFORE ROW, 212, 292
BEFORE STATEMENT, 212, 292
begin transaction, 56, 60, 92, 179
BEGIN TRANSACTION, 62, 147
BFILE, 227, 256
BIGDATETIME, 35, 227
BINARY, 35, 150, 226, 227
Bitmap, 66, 67
BLOB, 32, 36, 227, 271
BOOLEAN, 150, 232

C Applications, 126, 133
cache, 20, 90, 100, 104, 154, 187, 188, 203
CASE, 235, 236
CEIL, 234
CEILING, 234
chained, 60, 92, 178, 179, 181
CHAR, 35, 36, 70, 71, 80, 134, 161, 162, 163, 173, 177,
226, 234, 241, 242, 243, 244, 271
CHECK, 120, 232
checkalloc, 92, 189, 287
checkcatalog, 92, 189, 287
checkdb, 92, 189, 287
CHECKPOINT, 57
checkstorage, 69, 92, 189, 287
checktable, 189, 190, 287
checkverify, 189
CHR, 234
CIS, 13, 16, 66, 80, 81, 115, 116, 117, 194, 272, 274
Client Library, 132, 203, 204
CLOB, 32, 35, 227, 271

- CLOSE, 127, 177
- clustered index, 45, 65, 66, 191, 250, 251, 253, 283
- COBOL, 126
- COLUMN, 33, 48
- COMMIT, 56, 127, 147, 148, 178, 179, 180, 182
- commit transaction**, 60
- COMPUTE, 48
- Concurrency, 26, 53, 202
- CONNECT, 46, 47, 71, 73, 128
- Connecting to Oracle, 19
- CONSTANTS, 228
- constraints, 30, 31, 33, 44, 114, 120, 125, 184, 232, 251, 254, 256
- Constraints, 33, 119, 120
- Control Files, 37, 41
- CONVERT, 234, 235, 241, 242, 243, 244
- CREATE [OR REPLACE] FUNCTION, 156
- create database, 43, 109, 112, 113, 140
- create encryption key**, 193
- create function, 156, 158, 160
- create index, 77, 110, 112
- CREATE pfile, 103
- CREATE PROCEDURE, 123, 152, 153, 158, 161, 213, 230
- CREATE SEQUENCE, 122, 213, 230
- create table, 61, 66, 110, 112, 122, 193, 213
- CREATE TABLE, 33, 39, 76, 80, 122, 213, 230, 232
- Ct-Library, 133, 134, 136
- CURSOR, 160, 177, 237
- cursors, 157, 160, 176, 177, 178, 199, 246, 247, 248, 249, 251
- Data Block**, 39, 141
- Data Dictionary, 68, 69, 70
- Data Files**, 38
- Data Guard, 15
- data integrity, 28, 114, 125, 176, 184
- Data Pump**, 83, 276
- Data segment, 39
- Data Sources, 298, 300
- Data Storage, 36, 38
- data transfer, 79, 125
- Data Types, 32, 33, 34, 150, 271
- Database Device, 108
- Database Devices**, 38
- Database Links**, 123
- Database Security, 57
- datachange, 200, 286
- Datafiles, 36
- DataManager, 130
- Datapages**, 185
- datarow, 53, 54, 62
- Datarow**, 53, 186
- Datarows**, 185
- datatype, 13, 120, 157, 226, 228, 232, 250, 255, 256, 261
- datatypes, 80, 116, 151, 226, 250, 251, 255, 269
- DATE, 32, 35, 74, 80, 226, 227, 234, 235, 242
- DATETIME, 32, 35, 226, 242, 244
- dbca, 95
- dbcc**, 69, 88, 92, 157, 189, 190, 202, 252
- DBCC**, 189, 286, 287, 288
- dbccdb*, 69, 189
- dbo**, 109, 268, 269
- dbstart, 96
- DDL, 53, 76, 178, 182
- deadlock, 26, 55, 148
- deadlocks, 26, 28, 54, 77, 148, 203
- DECIMAL, 34, 226
- DECLARE, 123, 128, 148, 162, 163, 165, 168, 169, 170, 171, 173, 177, 213, 230, 242, 244
- DECLARE CURSOR, 128
- DECODE, 73, 236
- Default segment, 40
- defncopy, 95, 276
- DELETE, 52, 53, 54, 76, 168, 169, 176
- DESC, 73, 269, 293
- DESCRIBE INPUT, 128
- DESCRIBE OUTPUT, 128
- Dirty Read**, 27
- disk init, 108, 112
- DISTINCT, 48, 245, 253
- DML, 53, 76

DOUBLE, 34, 226
dsedit, 95
DSQUERY, 134, 261
DTM, 103
DUAL, 49, 163, 164, 168, 169
dump, 41, 59, 63, 64, 80, 88, 90, 93, 100, 157, 192, 193,
194, 277, 280, 281
dump database, 59, 80, 88, 90, 194
Dump Devices, 41
dump transaction, 63, 64, 90
DUP_VAL_ON_INDEX, 239
Dynamic Performance Views, 68, 73

ECDA, 81, 82, 117, 118, 223
embedded SQL, 126, 301
Embedded SQL, 126, 127
emca, 95
Encrypted Columns, 192
Encryption, 192, 193
Error Handling, 148, 239
EXCEPTION, 148, 170, 239
Exception-Handling, 181
EXEC, 123, 155, 156, 167, 213, 230, 242
EXECUTE, 128, 155, 159, 167, 231
EXECUTE IMMEDIATE, 128
EXIT, 168, 169, 229, 238, 241
exp, 95, 276
expdp, 95
export, 82, 83, 99, 115
Export, 276
ExpressConnect, 13
Extent, 39, 54
External Tables, 80

FETCH, 51, 128, 177, 229
flashback, 48, 63, 64, 279, 280, 281
Flashback, 48, 215, 279
FLOAT, 34, 35, 177, 226
FOR, 48, 51, 175, 177, 242
for update, 120, 249
forceplan, 296

full text search, 102
Full-Text Search, 101, 194, 195, 267
Function, 30, 67, 156, 159, 160

GETDATE, 235, 242
Global Variables, 181
GO, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170,
171, 173, 175, 176, 177, 228
GOTO, 170, 238
grant, 27, 58, 61, 84, 85, 86, 87, 109, 193
GROUP BY, 48, 236
guest, 269

Hash, 65, 111
Heap memory, 187
hint, 74, 124, 126, 293, 294, 295
HINT, 216, 293
Hints, 293
Historical Server, 25, 196, 197, 198, 199, 203, 204
HOLDLOCK, 54, 55, 179

identity, 30, 120, 122
IDENTITY, 250
IF, 148, 159, 162, 163, 164, 165, 168, 169, 170, 229,
233, 238, 253
IMAGE, 32, 35, 36, 150, 227, 256
imp, 95, 276
impdp, 95
import, 82, 99, 131, 262
Import, 95, 262, 276
Index Maintenance, 283
Index Rebuilds, 283
Index segment, 39
indexalloc, 189, 287
indexes, 30, 36, 42, 44, 66, 67, 76, 77, 79, 80, 81, 104,
110, 113, 115, 184, 190, 195, 202, 250, 251, 252, 254,
263, 264
INITCAP, 73, 229
Input recording, 197
insensitive, 177, 178, 199

INSERT, 48, 49, 53, 54, 76, 122, 123, 176, 213, 230, 239
 INSERT INTO, 48, 122, 123, 176, 213, 230
 Installation, 101, 102
Instance, 18, 84, 94, 99
 Instances, 18
 INT, 177, 226, 234, 241, 244
 ISNULL, 235
 isolation level, 27, 28, 29, 54, 60, 92, 179
 Isolation Levels, 27
 isql, 83, 88, 92

 Java, 11, 45, 70, 104, 131, 157, 183, 224
 JDBC, 104, 126, 129, 130, 131, 199, 297, 298, 299, 300
 JDBC drivers, 297
Job Scheduler, 25, 44, 101, 102, 200, 205, 206, 267, 277
 Join Orders, 296

 LAST_DAY, 235
 LD_LIBRARY_PATH, 134
 LENGTH, 234
 License, 101
 load database, 90
 load transaction, 91
 Loading, 32, 77, 79, 272
lock, 28, 48, 53, 54, 65, 71, 93, 106, 185, 203, 250
 Locking, 53, 185, 248
Log Devices, 40
 Log segment, 40, 110
 LOG_MINER, 281
 Logical Transaction, 56, 147
 LONG, 227
 LOOP, 168, 169, 238, 241, 242
 LTRIM, 241

 master, 41, 43, 44, 69, 73, 74, 88, 89, 91, 101, 102, 106, 107, 109, 231, 260
Master Database, 41, 69
 Materialized Views, 213
 max memory, 20, 21, 22, 100

MDA, 10, 70, 73, 74
 memory, 18, 19, 20, 21, 22, 58, 68, 90, 100, 102, 104, 105, 187, 188, 210, 224, 246, 290
 Memory, 15, 20, 22, 104, 187
 Memory Structure, 20
 Migrating the Application, 13
 Migrating the Data, 13, 114
 Migration Process, 12
 model, 43, 48, 91, 99, 107, 110, 115, 141, 145, 147, 178, 189, 201, 246, 255
 MONEY, 34, 226
 Monitor, 25, 94, 101, 102, 196, 202, 203, 204
Monitor Server, 25, 101, 102, 196, 203, 204, 267, 291
 Monitoring, 70, 104, 202, 210, 289
 monProcessObject, 73
 monProcessSQLText, 74
 multi-threaded, 23, 25
 multi-versioning, 26, 27

 netca, 95
 NO_DATA_FOUND, 239
NOARCHIVELOG, 277, 278
 nonclustered, 251
Nonrepeatable Read, 27
 NOT NULL, 120, 233
 NULL, 71, 73, 134, 223, 233, 235, 244, 261
 NUMBER, 34, 35, 148, 158, 161, 162, 163, 164, 165, 168, 169, 170, 171, 175, 177, 226, 232
 NUMERIC, 34, 226, 232, 234
 NVL, 70, 71, 72, 235

 OBJECT_ID, 220
 OBJECT_VALUE, 220
 OCI, 131, 133, 134, 136
 ODBC, 126, 129, 254, 300
 OEM, 95, 276, 289, 291
 OLAP, 15
 OLTP, 46, 74, 76, 100, 187, 251
 online database, 91
 OPEN CURSOR, 128
 Oracle forms, 126, 139, 145

Oracle Forms, 139, 140, 141, 145
Oracle RAC, 15
Oracle Web Access, 221
ORDER BY, 70, 72, 73, 74, 236, 245

Package, 119, 160
Package Body, 160
parallel, 74, 75, 76, 77, 78, 79, 106, 111, 113, 191,
203, 264

Parallel Data Loading, 77

Parallel DDL, 76

Parallel DML, 76

Parallel Execution, 74

Parallel Query, 74, 191

Parallel Recovery, 77

parallelism, 74, 75, 76, 77, 106, 191

Parallelism, 74, 294, 295

Partition, 111, 190

Password, 59, 194

performance, 13, 14, 22, 25, 36, 38, 42, 53, 54, 66, 67,
68, 73, 74, 76, 89, 92, 94, 99, 100, 103, 106, 109, 110,
111, 113, 114, 115, 154, 179, 180, 183, 184, 188, 191,
195, 196, 198, 202, 203, 204, 238, 246, 248, 249, 250,
251, 252, 264

PGA, 20

Phantom Read, 27

PL/SQL, 30, 36, 82, 83, 140, 147, 148, 150, 152, 153,
154, 155, 156, 159, 160, 161, 165, 166, 171, 177, 178,
181, 182, 257, 265, 269, 271, 272, 274, 275, 292

plan_table, 73

Planning, 114

Playback, 196, 197

PowerBuilder, 139, 140, 142, 143, 144, 145, 229

PowerDesigner, 13, 115, 124, 144, 254, 269, 271

PREPARE, 128

PRIMARY KEY, 120

PRINT, 168, 229, 240

Proxy Tables, 80

pubs2, 44

pubs3, 44

Query Plan, 70

Query Plans, 70

RAC, 18

RAISE_APPLICATION_ERROR, 240

RAISE_APPLICATION_ERROR, 181

raiserror, 149

RAISERROR, 166, 181, 229, 240

RAW, 35, 227, 271

Read Committed, 27

Read consistency, 54, 55

READ ONLY, 54, 55

Read Uncommitted, 27

REAL, 226

RECORD, 150, 229, 230

recovery, 14, 18, 39, 41, 44, 63, 64, 69, 77, 78, 88, 89,
91, 100, 106, 192, 201, 203

Recovery, 77, 103

Redo, 37, 40, 42, 63

Redo Log, 37, 40, 63

Redo Log Files, 37

REF_CURSOR, 292

REFCURSOR, 172, 175, 292

REFERENCE KEY, 120

Remote Server, 119

Reorganizing, 93

Repeatable Read, 28

Replication Server, 13, 15, 105, 223, 272, 274

Resource, 59, 247

Restore, 88

RETURN, 154, 156, 158, 159, 161, 162, 165, 173, 181,
221, 229, 232

revoke, 58, 84, 86, 87

RMAN, 276

Roles, 45, 58, 69, 84, 85, 188

ROLLBACK, 56, 64, 128, 147, 148, 178, 179, 180

Rollback segment, 39, 64

rollback transaction, 60, 65, 179

Round-robin, 111

ROWID, 35, 216

ROWNUM, 217
RTRIM, 241, 242, 243
rule, 22, 64, 120, 143

sa, 38, 188, 200, 223, 268
SAVEPOINT, 56, 178, 180, 181
scalar_expression, 157, 159
Schema, 30, 31, 152, 270
Scrollable Cursors, 199
Security, 57, 58, 84, 85, 105, 184, 193
Segments, 39, 40, 110
SELECT, 47, 48, 49, 51, 53, 54, 55, 68, 70, 71, 72, 73, 74, 76, 103, 123, 148, 158, 162, 163, 164, 168, 169, 170, 171, 172, 173, 175, 176, 177, 181, 213, 228, 229, 230, 231, 236, 241, 242, 244, 245, 253
semi_sensitive, 177
SEQUENCE, 230
Sequences, 31, 122, 213
Serializable, 28, 29
SERIALIZATION, 54
Server processes, 23, 100
set cursor rows, 249
set forceplan, 251
set role, 85
SET ROLE, 46, 47
set showplan, 202
set transaction, 179
SGA, 20
shared lock, 26, 54, 55, 148
shared memory, 18, 20, 21, 100
showplan, 92, 252
showserver, 97
single process, 20, 23, 25
Slave processes, 23
SMALLDATETIME, 227
SMALLINT, 34, 226
SMALLMONEY, 226
sp_addremotelogin, 123
sp_addsegment, 112
sp_addserver, 80, 123
sp_adduser, 84, 123
sp_bindexclass, 188
sp_changedbowner, 109
sp_configure, 71, 89, 92, 103, 106
sp_dboption, 79, 80, 92, 106, 107, 110, 115, 182, 263
sp_dbrecovery_order, 91
sp_dropsegment, 112
sp_help, 70
sp_helpdb, 61, 71, 72, 89
sp_helpdevice, 72, 89
sp_helpsegment, 89, 94
SP_LOCK, 54
sp_objectstats, 202, 224
sp_procxmode, 60
sp_server_info, 71
sp_showplan, 73
sp_spaceused, 94
sp_sysmon, 106, 203, 224
sp_tempdb, 200
sp_who, 70, 188
SQL language, 11, 13, 46, 123, 124, 126, 224, 265, 292
SQL Language, 46
SQL%ROWCOUNT, 170, 171, 182
SQL*Net, 132, 265
SQL*Plus, 94, 96, 276
SQLCA, 128
SQLCODE, 128, 170, 171, 181, 239
SQLLDR, 79
SQLSTATE, 128
startserver, 96
statistics, 77, 92, 94, 190, 200, 202, 203, 206, 223, 248, 251
Step by Step Migration Example, 265
stored functions, 44
stored procedures, 25, 30, 43, 44, 58, 69, 70, 73, 74, 80, 88, 94, 104, 126, 130, 146, 147, 153, 154, 160, 161, 167, 174, 184, 191, 204, 205, 229, 230, 231, 232, 235, 251
Stored Procedures, 13, 60, 146, 147, 229, 231, 293
SUBSTR, 234, 241
SUBSTRING, 234, 241, 242
Sybase ASE Cluster Edition, 15, 18, 30

Sybase Backup Server, 25

Sybase Central, 94, 95, 109, 203, 291

Sybase Control Center, 95, 204, 207, 208, 209, 291

Sybase Historical Server, 291

Sybase IQ, 15, 32, 48, 80, 83

Sybase manuals, 267

Sybase Mirror Activator, 15

Sybase Unified Agent, 18, 25

Sybase XP Server, 25

sybmgtmdb, 44, 205

sybsystemdb, 43, 69, 91

sybsystemprocs, 43, 91

synonym, 230

Synonyms, 31, 122, 212

SySAM, 266

SYSDATE, 235

sysdba, 96

syslogins, 89, 268

syslogs, 90, 187

sysprocesses, 62, 231

System segment, 40

System Views, 68

SYSTIMESTAMP, 235

TABLE, 33, 48, 53, 68, 70, 76, 150, 229

tablealloc, 189, 287

tables, 30, 36, 39, 40, 41, 43, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 58, 62, 63, 65, 66, 67, 68, 69, 70, 73, 74, 76, 77, 79, 80, 81, 83, 84, 92, 104, 110, 111, 113, 115, 122, 125, 146, 150, 152, 176, 177, 179, 184, 186, 187, 189, 190, 197, 200, 201, 202, 205, 224, 231, 246, 249, 250, 251, 254, 256, 262, 263, 264

Tablespaces, 39

TDS, 265, 300

tempdb, 40, 43, 66, 91, 200, 201, 224

Temporary Database, 200

Temporary segment, 39

TEXT, 32, 35, 36, 102, 150, 227

textalloc, 287

Threshold, 93

TIME, 227

TIMESTAMP, 32, 35, 227, 235

TINYINT, 34, 226

TO_CHAR, 234, 241

TO_NUMBER, 234

TOO_MANY_ROWS, 239

traceflags, 202

traceon, 202, 252

Transaction Processing, 60

transactions, 14, 20, 27, 28, 39, 40, 41, 42, 43, 44, 53, 54, 55, 56, 57, 59, 60, 61, 62, 63, 64, 74, 90, 106, 147, 148, 176, 178, 179, 180, 181

Transactions, 62

triggers, 13, 30, 31, 44, 70, 120, 126, 146, 184, 191, 224, 251, 254, 256, 263, 264

Triggers, 13, 31, 119, 120, 124, 146, 232, 292

TRUNC, 234, 235

T-SQL, 30, 56, 109, 131, 181, 182, 229, 237, 238, 257, 269, 271, 286, 292

Tuning, 104, 105, 124, 126, 250

UGA, 20

Undo, 39, 63, 64

Unicode, 105

Unified Agent, 18, 101, 102, 207, 268

UNION, 231, 245, 253

union all, 248

UNIQUE, 120, 221

Unloading, 79, 82, 272

until_time, 215, 281

UPDATE, 48, 51, 53, 54, 76, 123, 164, 178, 213, 230

Update statistics, 284, 286

Update Statistics, 200, 284

user accounts, 84

user connections, 19

user-defined functions, 159, 231

user-defined stored procedure, 231

userenv, 231

users, 19, 30, 39, 43, 44, 48, 53, 55, 57, 58, 60, 68, 83, 84, 85, 86, 87, 88, 99, 101, 108, 139, 143, 158, 183, 184, 186, 187, 188, 192, 196, 203, 223

v\$database, 71
v\$datafile, 72
v\$instance, 71
v\$logfile, 72
v\$process, 70
v\$session, 70, 71
v\$sga, 71
v\$sqlarea, 73
v\$sqltext_with_newlines, 74
v\$version, 71
Validate Migration, 14, 224
Validation, 114, 145, 183
VALUES, 50, 76, 122, 123, 213, 230
VARBINARY, 227

VARCHAR, 35, 226
VARCHAR2, 32, 35, 49, 226, 271
versioning, 53, 176
Versioning, 26
views, 30, 31, 44, 47, 50, 51, 52, 55, 58, 68, 70, 74, 81,
82, 119, 120, 122, 196, 198, 254

Web Services, 25, 101
WHEN, 148, 170, 229, 235, 236, 239, 241
WHILE, 168, 169, 238, 241, 242
with recompile, 154
worker processes, 75, 106, 203

XP Server, 101, 102, 267