SYBASE®

Migration Technology Guide

Adaptive Server® Enterprise

15.0.3

DOCUMENT ID: DC00967-01-1503-01

LAST REVISED: March 2009

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This book is intended for System Administrators who are migrating to a different version of Adaptive Server Enterprise.

How to use this book

Chapter 1, "Migration Strategy," outlines strategies for users who want to upgrade and use new features immediately, and for users who would like to upgrade now and use the new features later.

Chapter 2, "QPTune," gives a comprehensive summary of the QPTune tool.

Chapter 3, "Running the Query Processor in Compatibility Mode," discusses "the compatibility mode" feature for users who want to upgrade to a new version, but retain performance characteristics of a previous version.

Related documents

The Adaptive Server® Enterprise documentation set consists of:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Product Manuals Web site.

- The *Installation Guide* for your platform – describes installation, upgrade, and some configuration procedures for all Adaptive Server and related Sybase products.

- *New Features Guide* – describes the new features in Adaptive Server version 15.0.3, the system changes added to support those features, and changes that may affect your existing applications.

- *ASE Replicator Users Guide* – describes how to use the Adaptive Server Replicator feature to implement basic replication from a primary server to one or more remote Adaptive Servers.

- *Component Integration Services Users Guide* – explains how to use the Component Integration Services feature to connect remote Sybase and non-Sybase databases.

- The *Configuration Guide* for your platform – provides instructions for performing specific configuration tasks.
- *Enhanced Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server Users Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes, functions, and stored procedures in the Adaptive Server database.
- *Job Scheduler Users Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).
- *Messaging Service Users Guide* – describes how to use Real Time Messaging Services to integrate TIBCO Java Message Service and IBM WebSphere MQ messaging services with all Adaptive Server database applications.
- *Migration Technology Guide* – describes different strategies and tools for migrating to a different version of Adaptive Server.
- *Monitor Client Library Programmers Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server Users Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Performance and Tuning Series* – is a series of books that explain how to tune Adaptive Server for maximum performance:
  - *Basics* – contains the basics for understanding and investigating performance questions in Adaptive Server.
  - *Improving Performance with Statistical Analysis* – describes how Adaptive Server stores and displays statistics, and how to use the set statistics command to analyze server statistics.
  - *Locking and Concurrency Control* – describes how to use locking schemes to improve performance, and how to select indexes to minimize concurrency.

    - *Monitoring Adaptive Server with sp_sysmon* – describes how to use sp_sysmon to monitor performance.
    - *Monitoring Tables* – describes how to query Adaptive Server monitoring tables for statistical and diagnostic information.
    - *Physical Database Tuning* – describes how to manage physical data placement, space allocated for data, and the temporary databases.
    - *Query Processing and Abstract Plans* – describes how the optimizer processes queries and how to use abstract plans to change some of the optimizer plans.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book (regular size when viewed in PDF format).
- *Reference Manual* – is a series of books with detailed Transact-SQL information:
    - *Building Blocks* – discusses datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
    - *Commands* – documents commands.
    - *Procedures* – includes system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.
    - *Tables* – discusses system tables and dbcc tables.
- *System Administration Guide* –
    - *Volume 1* – provides an introduction to the basics of system administration, including a description of configuration parameters, resource issues, character sets, sort orders, and instructions for diagnosing system problems. The second part of this book is an in-depth description of security administration.
    - *Volume 2* – includes instructions and guidelines for managing physical resources, mirroring devices, configuring memory and data caches, managing multiprocessor servers and user databases, mounting and unmounting databases, creating and using segments, using the reorg command, and checking database consistency. The second half of this book describes how to back up and restore system and user databases.

- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Full-size available only in print version; a compact version is available in PDF format.
- *Transact-SQL Users Guide* – documents Transact-SQL, the Sybase-enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the pubs2 and pubs3 sample databases.
- *Troubleshooting Series* (for 15.0) –
  - *Troubleshooting: Error Messages Advanced Resolutions* – contains troubleshooting procedures for problems that you may encounter. The problems addressed here are those the Sybase Technical Support staff hear about most often.
  - *Troubleshooting and Error Messages Guide* – contains detailed instructions on how to resolve the most frequently occurring Adaptive Server error messages. Most of the messages presented here contain error numbers (from the master..sysmessages table), but some error messages do not have error numbers, and occur only in the Adaptive Server error log.
- *Users Guide for Encrypted Columns* – describes how to configure and use encrypted columns with Adaptive Server.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Unified Agent and Agent Management Console* – describes the Unified Agent, which provides runtime services to manage, monitor and control distributed Sybase resources.
- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Web Services Users Guide* – explains how to configure, use, and troubleshoot Web Services for Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.

- *XML Services in Adaptive Server Enterprise* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that are available in XML services.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, there are links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.
3. In the Certification Report filter select a product, platform, and time frame and then click Go.

4 Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following sections describe conventions used in this manual.

SQL is a free-form language. There are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and most syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented. Complex commands are formatted using modified Backus Naur Form (BNF) notation.

Table 1 shows the conventions for syntax statements that appear in this manual:

Table 1: Font and syntax conventions for this manual

Element	Example
Command names, procedure names, utility names, and other keywords display in sans serif font.	select sp_configure
Database names and datatypes are in sans serif font.	master database
Book names, file names, variables, and path names are in italics.	*System Administration Guide* *sql.ini* file *column_name* *$SYBASE/ASE* directory

Variables—or words that stand for values that you fill in—when they are part of a query or statement, are in italics in Courier font.	`select column_name`
	`from table_name`
	`where search_conditions`

Type parentheses as part of the command.	`compute row_aggregate (column_name)`
Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates “is defined as”.	`::=`
Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces.	`{cash, check, credit}`
Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets.	`[cash | check | credit]`
The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command.	`cash, check, credit`
The pipe or vertical bar (|) means you may select only one of the options shown.	`cash | check | credit`

An ellipsis (...) means that you can *repeat* the last unit as many times as you like.	`buy thing = price [cash | check | credit]`
	`[, thing = price [cash | check | credit]]...`
	You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  sp_dropdevice [*device_name*]

  For a command with more options:

  select *column_name*
  from *table_name*
  where *search_conditions*

  In syntax statements, keywords (commands) are in normal font and identifiers are in lowercase. Italic font shows user-supplied words.

- Examples showing the use of Transact-SQL commands are printed like this:

  ```
  select * from publishers
  ```

- Examples of output from the computer appear as follows:

```
pub_id    pub_name                city          state
-------   ---------------------   -----------   -----
0736      New Age Books           Boston        MA
0877      Binnet & Hardley        Washington    DC
1389      Algodata Infosystems    Berkeley      CA

(3 rows affected)
```

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, SELECT, Select, and select are the same.

Adaptive Server sensitivity to the case of database objects, such as table names, depends on the sort order installed on Adaptive Server. You can change case sensitivity for single-byte character sets by reconfiguring the Adaptive Server sort order. For more information, see the *System Administration Guide*.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Adaptive Server HTML documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Migration Strategy

Sybase Adaptive Server Enterprise includes a sophisticated query optimizer that analyzes statistics from queried tables, using advanced algorithms to provide better performance. Adaptive Server 15.0.3 ESD #1 and later include tools that support more effective use of Adaptive Server 15.0's advanced query optimizer.

This book discusses optimization goals and recommendations for upgrading from Adaptive Server 12.5 to Adaptive Server 15, analyzing performance differences between the two versions, and tuning Adaptive Server 15.0 installations.

This book also documents a tool called QPTune, which enables users to identify and apply the best query plan, optimization goals, and other configuration settings.

Preupgrade considerations

This section covers optimization goals and criteria, and the steps you must complete before you upgrade to Adaptive Server 15.0. It includes a summary of preupgrade tests that help you evaluate performance after upgrading your production server.

Understanding optimization goals

A central concept of Adaptive Server 15.0's query processing engine is the "optimization goal", which provides an indication of the nature of the query being optimized. The Adaptive Server query optimizer determines how best to optimize a query based on optimization goals.

For example, a typical OLTP (online transaction processing) query and a typical DSS (decision-support system) query result in very different query plans due to the different data access patterns used by these queries. OLTP queries generally affect only one or a few rows and join only a few well-indexed tables. However, DSS queries typically affect many rows, return a few rows, and may join many tables.

Because of their different access patterns, OLTP queries often run most efficiently using a classic "nested-loop join", whereas DSS queries are more likely to run faster with a "hash join". If you indicate that a query is for OLTP or DSS purposes, the optimizer uses that information to generate a query plan that may save time, memory, and CPU usage.

Adaptive Server 15.0 provides three optimization goals, ordered from "narrow" to "wide," which correspond to the number of options and strategies that they allow the optimizer to consider:

- allrows_oltp – is best for OLTP queries. allrows_oltp offers the narrowest selection of join methods: the query optimizer considers only nested-loop joins.
- allrows_mix – is the default after upgrading to Adaptive Server 15.0. allrows_mix allows the optimizer to consider merge joins as well as parallel plans (if the Adaptive Server is configured for parallelism).
- allrows_dss – is best for DSS queries. allrows_dss offers the widest selection of join methods. The optimizer considers hash joins, as well as nested-loop joins, merge joins, and parallel plans.

If you use allrows_mix and allrows_dss, additional low-level processing algorithms are enabled for SQL operations; these algorithms are disabled if you use allrows_oltp.

When you widen the optimization goal, the query optimizer might use significantly more resources (time and procedure cache) to generate a query plan. If the optimizer generates the same query plan, with only nested-loop joins, under allrows_dss and allrows_oltp, you may expect the optimization under allrows_dss to take more time and procedure cache than under allrows_oltp.

The choice of optimization goal can have a significant impact on query performance. If you know that a certain application has different workload characteristics than the rest of your system, you may want to set an appropriate session-level optimization goal for that application. Either use the QPTune utility, or manually experiment with different optimization goals, and select one that provides the best overall performance for your particular set of applications and queries. See Chapter 2, “QPTune” for more details.

You can define the optimization goal at the server-, session- or individual-query level:

- Server-wide default:

```
sp_configure 'optimization goal', 0, 'allrows_dss'
```

- Session-level setting (overrides server-wide setting):

```
set plan optgoal allrows_dss
```

- Query-level setting (overrides server-wide and session-level settings):

```
select * from T1, T2 where T1.a = T2.b
plan '(use optgoal allrows_dss)'
```

Note You can also use a login trigger to set the session-level optimization goal.

Optimization criteria

An optimization goal is a collection of “on/off” settings for a series of properties known as “optimization criteria.” Optimization criteria allow or disallow the optimizer to consider a particular algorithm for access methods, joins, grouping, sorting, and so on.

For example, to enable hash joins, use the optimization criterion:

```
set hash_join on
```

Or, to disable the “store_index” algorithm (reformatting), use the optimization criterion:

```
set store_index off
```

The optimizer may decide to ignore a given criteria or goal for semantic reasons. For example, if a user disables all join operators, the new optimizer enables "nested loop" automatically.

Note Sybase recommends that you use optimization goals, instead of explicit settings for optimization criteria, unless advised to do otherwise by Sybase Technical Support.

Parallel query processing in Adaptive Server 15.0

Since version 11.5, Adaptive Server has supported parallelism within queries, whereby a single query is processed by multiple worker processes. You can use parallelism to improve response times for DSS-type queries, where a large number of rows are accessed, but only a small result set is returned.

Since the query processing features in Adaptive Server 15.0 offer potential performance benefits for DSS-type queries, Sybase recommends that, when you upgrade to Adaptive Server 15.0, you do not initially use parallel processing.

Serial processing is more resource-efficient than parallel processing although parallel processing allows you to deliver better overall performance with the same hardware. Also, Adaptive Server 15.0 in serial mode runs queries faster than earlier versions of Adaptive Server with parallelism.

However, parallelism may deliver better response times than serial processing for queries that use semantic table partitioning in Adaptive Server 15.0, or for DDL commands such as create index.

Resource recommendations for Adaptive Server 15.0

Adaptive Server 15.0 requires more procedure cache than version 12.5. This increased memory requirement applies to optimization as well as to query execution. Sybase recommends that you increase your procedure cache 2 – 6 times the size of your procedure cache in Adaptive Server 12.5.

You may also need to increase the space on tempdb for query processing on Adaptive Server 15.0.

Incorporating statistics in Adaptive Server 15.0

Adaptive Server uses a cost-based query optimizer to choose the best plan for a particular query. The optimizer estimates the cost of different plans based on statistics about the tables, indexes, partitions, and columns referenced in a query. Cost is computed in terms of I/O and CPU time. The optimizer then chooses the query plan method that has the lowest cost. Inaccurate statistics lead to incorrect cost estimates, and may result in a suboptimal choice of plans and reduced performance.

Some statistics, such as the number of pages or rows in a table (stored in systabstats), are updated automatically during query processing. Other statistics are updated only when update statistics runs, or when indexes are created. Examples of this are the histograms on column and density information, stored in sysstatistics.

Adaptive Server 15.0 is more susceptible to incorrect statistical data than earlier Adaptive Server versions, because multiple algorithms are used for sorting, grouping, unions, joins, and other operations. In addition, Adaptive Server 15.0 uses statistics in more ways than in Adaptive Server 12.x. For example, Adaptive Server 15.0 uses statistics to determine the join order in multitable queries.

Sybase recommends that you maintain up-to-date histograms for all columns referenced in where clauses, both when the where clauses are used as join predicates and as search arguments. Use the statistics advisor in QPTune to identify critical and missing statistics.

Recommended testing before upgrade

Before upgrading your production systems to Adaptive Server 15.0, gather details about the performance characteristics of your applications in the production environment of the current, pre-15 version of Adaptive Server. Gathering such data provides a statistical basis for performance analysis.

To compare Adaptive Server 12.x and 15.0 performance, run:

- Tests for as many application functions as possible, especially the most critical ones. For each function, measure the response time or throughput. If possible, perform these measurements for each query executed by the application.
- Performance measurements in your current Adaptive Server 12.x production system.

- The same function and performance measurement tests in a fully configured “test” system running Adaptive Server 15.0, with a copy of the full Adaptive Server 12.x production database, and a realistic workload. Run the same queries as in Adaptive Server 12.x, and with the same level of concurrent user activity. Capturing the “performance footprint” of your current Adaptive Server 12.x production environment provides a good baseline for any comparisons with Adaptive Server 15.0. The measurements you capture should include the number of logical I/O operations, elapsed time, compilation time, CPU utilization, showplan output, and so on. To enable a sensible comparison of performance in Adaptive Server 12.x and Adaptive Server 15.0, gather performance data at two levels, from:
  - Individual queries in isolation, and with a full workload run by multiple users
  - Adaptive Server as a whole, from a server-wide resource usage perspective

Several critical aspects affect performance numbers between Adaptive Server 12.x and 15. To avoid misleading performance numbers:

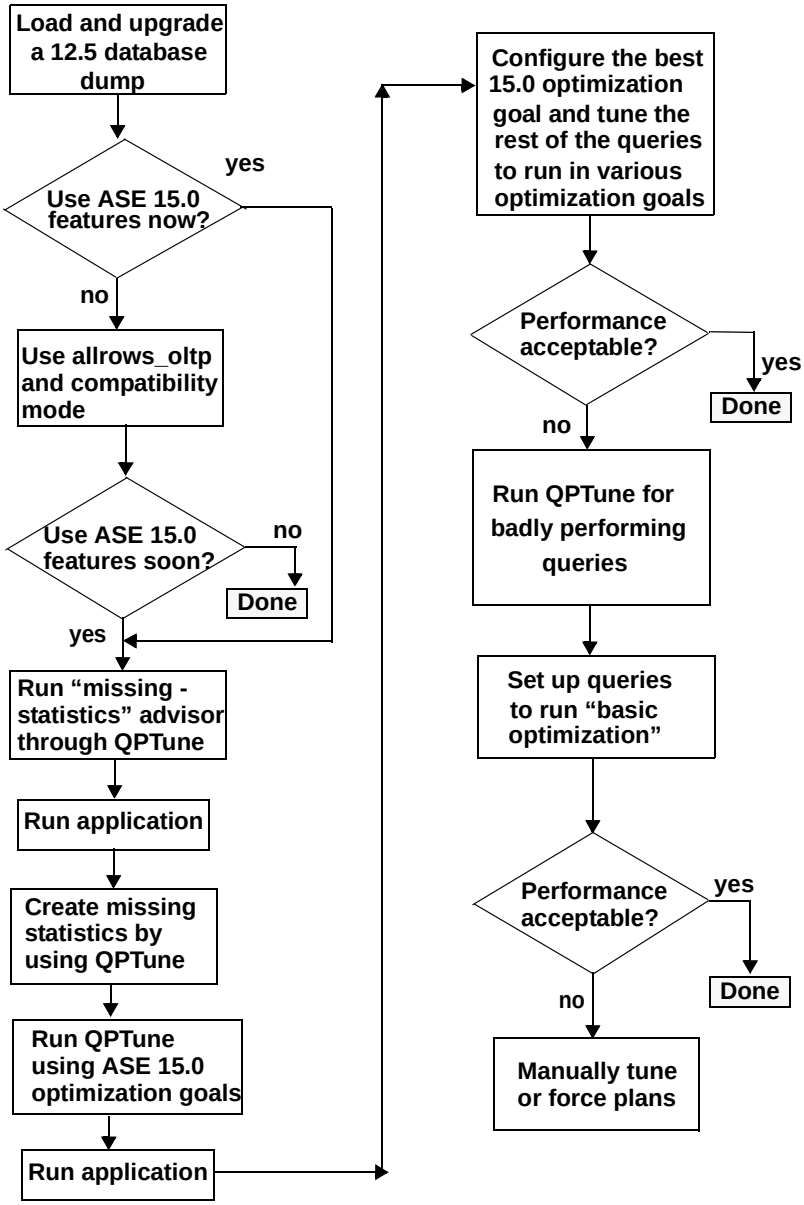
- “Warm up” the cache in the same manner for both Adaptive Server 12.x and Adaptive Server 15.0 testing.
- Use identical cache/buffer pool configurations.
- Increase the procedure cache in Adaptive Server 15.0 to about 2 – 6 times the amount used in Adaptive Server 12.x.
- Use similar data device layout and placement, especially for log devices and for tempdb.
- Set up test systems where you can easily restore the original database after each test run, especially when data is modified during testing.

Note You may need to increase the size of the data cache in Adaptive Server 15.0

Migrating to Adaptive Server 15.0 features

You may want to use Adaptive Server 15.0 features immediately after upgrade, or you may prefer to use new features later. The following flowchart depicts the different strategies available for Adaptive Server migration:

Figure 1-1: Flowchart of Adaptive Server migration strategy



Upgrading, and using new features immediately

To upgrade to Adaptive Server 15.0 and use the new features immediately, Sybase recommends that you skip setting the compatibility mode, and tune the application using QPTune:

1 Upgrade your Adaptive Server 12.5 and database to Adaptive Server 15.0.3 ESD #1 or later, which supports migration using QPTune.

2 Use QPTune to turn on the statistics advisor.

3 Run the application queries. QPTune advises you on what statistics are critical, and creates them. Typically, most queries are tuned at this point.

4 Run QPTune with Adaptive Server 15.0 optimization goals.

5 Run the queries with different optimization goals and select the best performing optimization goal.

6 Run the application queries again.

7 Check if there are any queries left to be tuned.

8 Run QPTune for queries that need further tuning.

9 Manually tune the remaining queries. Use traditional techniques of analyzing query plans and either rewriting them to obtain the desired performance, or using techniques such as abstract query plans.

Upgrading, and using new features later

If you are going to upgrade to Adaptive Server 15.0, and incrementally begin using Adaptive Server 15.0 features, upgrade your Adaptive Server 12.5 and database to Adaptive Server 15.0.3 ESD #1 or later, which supports migration using QPTune. Use allrows_oltp as the optimization goal and enable compatibility mode for the upgrade.

When you are ready to use Adaptive Server 15.0 features:

1 Use QPTune to turn on the statistics advisor.

2 Run the application queries. QPTune advises you on what statistics are critical, and creates them. Typically, most queries are tuned at this point.

3 Run QPTune using Adaptive Server 15.0 optimization goals.

4 Run the queries with different optimization goals and select the best performing optimization goal.

5 Run the application queries again.

6 Check if there are any queries left to be tuned.

7 Run QPTune for queries that need further tuning.

8 Manually tune the remaining queries using abstract query plans.

Note You can incrementally migrate stored procedures using the same methodology.

Upgrading, but not using new features

If you are upgrading from Adaptive Server 12.5, but are not going to use Adaptive Server 15.0 features, use allrows_oltp as an optimization goal and enable compatibility mode.

For more information on compatibility mode, see Chapter 3, "Running the Query Processor in Compatibility Mode."

Troubleshooting

This section discusses query processing performance, and strategies for addressing optimization problems.

Query processing tips

Adaptive Server 15.0 offers a greatly improved query processing environment. However, if query plans or query performance are not what you expect, here are some ways to isolate the problem:

- When using different optimization goals, make sure no cached plans are used: changing the session-level or server-wide optimization goal does not recompile cached plans. For stored procedures, either execute them with recompile, or run sp_recompile on one of the tables being accessed. For batches, make sure the statement cache is disabled by running set statement_cache off first.

- To ensure that a stored procedure is always optimized with a particular optimization goal, regardless of server-wide or session-level settings, use set plan optgoal allrows_xxx as the first statement in the stored procedure. This works only on Adaptive Server 15.0.2 ESD #2 or later.
- If your SQL code from Adaptive Server 12.x contains explicitly forced join orders (with set forceplan), reexamine the join orders before upgrading to Adaptive Server 15.0. Such constructs may prevent you from benefiting fully from the capabilities of Adaptive Server 15.0.

  With Adaptive Server 15.0.1 ESD #2 or later, you can enable two trace flags:

  - Trace flag15307 nullifies the effect of any set forceplan statements during query plan compilation.
  - Trace flag 15308 nullifies any explicit forcing of indexes, prefetch, parallelism, or buffer replacement strategies

  You can set both these trace flags (15307 and 15308) during server start-up, or dynamically enable them using dbcc traceon. The effects of both are server-wide and neither affects any query plan properties defined by abstract query plans.
- If your system consumes too much space in tempdb, use the Monitoring and Diagnostic Access tables to see if any particular session consumes a lot of space in a worktable. Enable the monitoring tables and run the following query:

  ```
  select SPID, DBName, ObjectName, PartitionSize
  from master..monProcessObject
  where DBID = tempdb_id(SPID)
  order by SPID
  ```

  Look for sessions that have a large value for PartitionSize. Worktables have an ObjectName of “temp worktable.” Find the corresponding SQL statement for the sessions by issuing queries to monProcessSQLText or monProcessStatement in the master database.

  To stop sessions from filling up tempdb, and thus affecting other sessions also requiring tempdb space, create a resource limit of type “tempdb_space.” You may also create multiple temporary databases and assign them to specific users. To check the tempdb space used by a single session, use:

  ```
  select pssinfo(spid|0,'tempdb_pages')
  ```

- Enable the statement cache and literal autoparameterization settings while running large numbers of identical or similar client-generated SQL queries in Adaptive Serer 15.0.1 or later. This does not include stored procedures, or execute-immediate query forms, and the queries may differ only in their search parameters. Overall performance is improved by significantly reducing the time and resources spent on query optimization.

  When the statement cache is enabled, a query's plan is cached so you need not compile an identical query, and thereby save time and resources. The statement cache is enabled server-wide with the configuration parameter statement cache size. At the session level, disable the statement cache with set statement_cache off.

  Literal autoparameterization is enabled server-wide with the configuration parameter enable literal autoparam, and at the session level with set literal_autoparam on. enable literal autoparam applies only when the statement cache is enabled. With literal autoparameterization enabled, caching is extended to almost-identical queries that differ only in a constant value. For example, these two queries are not considered identical:

  ```
  select CustName from Customers where CustID = 123
  select CustName from Customers where CustID = 456
  ```

  However, they are likely to generate the same query plan. Enabling literal autoparameterization has the effect that the statement cache factors out the constant value in the where clause and caches a plan for all queries that look like this:

  ```
  select CustName from Customers where CustID=
  <integer-constant>
  ```

Obsolete optimization commands in Adaptive Server 15.0

Various optimization-related settings from 12.x are no longer relevant in Adaptive Server 15.0. Although the following commands still exist in Adaptive Server 15.0, they are relevant only in compatibility mode, and do not have any effect on the Adaptive Server 15.0 optimization process:

- set sort_merge – this has been replaced by set merge_join, optimization goals and the configuration parameter enable merge join.
- set jtc – join transitive closure is always enabled in Adaptive Server 15.0.
- set table count – this setting is no longer relevant in Adaptive Server 15.0.
- enable sort-merge join and JTC – this configuration parameter has been replaced by optimization goals and by the configuration parameter enable merge join.

- Start-up trace flags 334 and 384 – these flags enabled merge joins and JTC and are no longer relevant.

Sybase recommends that you remove any references to these features from your applications.

Information to capture before contacting Technical Support

Before contacting Tech Support, gather as many diagnostics statistics as possible, especially when you can reproduce the problem.

701 errors

When a regular query (excluding update index statistics) generates a 701 error, it indicates that Adaptive Server has exhausted the procedure cache space. If you are running with the default procedure cache size, increase procedure cache and try again. The general guideline for version 15.0 and later is to use a procedure cache that is 2 – 6 times the size of your 12.5.x procedure cache. In some cases, especially while using the optimization goal allrows_dss, your procedure cache may need to be even larger.

If increasing the procedure cache does not resolve the 701 error and you cannot isolate the problem, set up a configurable shared memory dump that includes the procedure cache pages:

```
sp_configure 'dump on conditions', 1
go

sp_shmdumpconfig 'add', 'error', 701, 1,
'my_dump_directory',null,include_proc
go
```

sp_shmdumpconfig adds the error 701 condition to initiate a memory dump.The fourth parameter (1 in the examples above) indicates the number of memory dumps to capture. Adaptive Server does not capture additional memory dumps on this condition until Adaptive Server is restarted or until you manually reset the counter.

The parameter *my_dump_directory* is the name of a directory to hold the memory dump. The file system on which the directory resides should have enough free space to hold the memory dump file, which can be large. Verify the dump conditions currently defined by running sp_shmdumpconfig without any parameters. This also shows an estimated size of the memory dump to be captured.

The parameter *include_proc* enables procedure cache information to be included in the configurable shared memory dump.

A file name that includes the date and time of the memory dump is automatically generated. Once the memory dump has been captured, reset the system using:

```
sp_shmdumpconfig 'drop', 'error', 701
go
```

By default, Adaptive Server sends the 701 error message to the client. You may also have this message reported in the error log by running:

```
sp_altermessage 701,'with_log',true
```

To stop all configurable shared memory dumps, set dump on conditions to 0. Once the memory dump has been captured, open a case with Technical Support and upload the memory dump to the FTP site. Include the output from the SQL statements below which use the monitoring tables within Adaptive Server:

```
select * from master..monProcedureCacheMemoryUsage
select * from master..monProcedureCacheModuleUsage
go
```

The monitoring tables are automatically set up during execution of the installmaster script in Adaptive Server 15.0.2 or later. The installation process for earlier versions of Adaptive Server execute the installmontables script. See Adaptive Server 15.0 documentation for more details on configuring the monitoring tables.

Performance problems with a limited number of queries

If a limited number of queries are not performing well due to suboptimal query plans or suboptimal resource consumption, install the latest Adaptive Server 15.0.x version on your development server. If the problem still exists, submit a reproduction of the problem or diagnostics to Technical Support. To gather diagnostics:

1 Create a script file *sql.txt* containing these commands:

```
select @@version
go
select @@optgoal
go
sp_cacheconfig
go
sp_configure 'nondefault' (only if you're running
Adaptive Server 15.0.2 or later)
```

```
go
dbcc traceon(3604)
set showplan on
set statistics time, io, plancost on
set option show long
go
<your query text>
go
```

Note set option show long may produce a lot of output for complex queries.

2 Use isql to execute *sql.txt* and capture the output in a file:

```
isql –Usa –P yourpassword -S YOUR_SERVER_NAME
–i sql.txt –o sql.out
```

Use the -w option of isql to format the output.

3 Send this information to Technical Support:

- The *sql.txt* and *sql.out* files. If available, include the "fast" (*sql.fast.txt*) and "slow" (*sql.slow.txt*) query plans, and corresponding output files *sql.fast.out*, *sql.slow.out.*
- DDL for the base tables and indexes, which you can generate using the ddlgen utility.
- Simulate statistics output for the base tables using optdiag:

  ```
  optdiag statistics simulate <table-name>
  -Usa -P yourpassword -S YOUR_SERVER_NAME
  -o <output-file>
  ```

- A copy of the Adaptive Server configuration file. For Adaptive Server15.0.2, include the output of sp_configure 'nondefault'.
- If the query uses views or stored procedures, then include their SQL source code obtained using defncopy or ddlgen.
- The output of sp_monitorconfig 'all' and sp_helpsort.

System-wide performance issues

If the performance of Adaptive Server at the server level is not acceptable, and you are running 15.0.2 ESD #3 or later, you may shut down the Adaptive Server and restart it with trace flag 757 set in the RUN_server file. This is also effective when you experience unusually high levels of CPU usage without a clear cause, while running a multiengine Adaptive Server.

If the procedure cache is filled with idle cached plans, and the CPU usage is not high, run the following dbcc commands instead. However, using these commands is likely to have a lesser effect than restarting the server.

```
dbcc traceon(757)
go
dbcc proc_cache(free_unused)
go
```

Note Do not use trace flag 757 in Adaptive Server versions earlier than 15.0.2 ESD #3.

Uploading diagnostics to Technical Support

After you have created diagnostic files, open a case with Technical Support. To upload diagnostics to the Sybase FTP site, contact Technical Support for current instructions.

CHAPTER 2

QPTune

Topic	Page
Setting up your system	18
Using QPTune to fix missing statistics	19
Using QPTune to tune queries or applications	24
Configuration file	32
Examples	34
Upgrade issues	43
Localization	44
QPTune GUI	44
QPTune reference information	50

QPTune is an Adaptive Server utility that is written in Java/XML. It enables users to identify the best query plan, optimization goals, or other configuration settings, and apply them at the server or query level. This results in optimal performance of subsequent query executions. Once you have identified the best settings for application queries, you can export and apply them to production servers.

QPTune allows users to:

- Fix missing statistics in an application
- Tune an application to find the best optimizer settings for any number of queries
- Selectively apply customized or standard settings to specific queries using user-defined rules

Use QPTune to analyze and compare any number of configuration settings or Adaptive Server installations to generate a performance impact analysis report, or to perform plan fixes without degrading Adaptive Server's performance. Adaptive Server gathers metrics with simple select statements and stores it in stored procedure or statement caches. Adaptive Server fixes query plans using DDL statements that have little impact on the overall performance of the system. In addition, QPTune allows different threshold levels for monitoring, thereby reducing the metrics that need to be collected.

Setting up your system

Before starting QPTune, set these environment variables:

- SYBASE_JRE6 and JAVA_HOME – to the Java runtime installation.
- SYBASE – to the latest Sybase installation on your machine.
- SYBASE_ASE – to the Adaptive Server component(directory) of the installation on your machine.

The QPTune executable is named *QPTune* on UNIX and *QPTune.bat* on Windows and is found in:

$SYBASE/$SYBASE_ASE/qptune, on UNIX
%SYBASE%\%SYBASE_ASE%\qptune, on Windows

For complete syntax and reference information on QPTune, see "QPTune reference information" on page 50.

To verify your environment and installation, and for information on basic syntax, run QPTune with the -h option:

```
QPTune –h
```

Sample output in a Windows environment:

```
QPTune <Version 3.0> Windows/Unix Built: Fri Jan 21 14:00:15 PDT 2009
Syntax:
QPTune [-U <username>] [-P <password>] [-S <hostname:port/database>]
[-A <action
[start|collect(_full)|compare|fix|(start|collect|fix|undo_fix)_stats]>]
[-M <mode>] [-T <appTime>] [-i <inputFile>] [-o <outputFile>]
[-f <fileList(,)>] [-c <configFile>] [-l <limit>] [-e <evalField>]
[-d <diff%(,diff_abs)>] [-m <missingCount>] [-n <login>] [-J <charset>]
[-N (noexec)] [-g (applyOptgoal)][-v (verbose)] [-s (sort)] [-h (help)]
Example:
QPTune -U sa -P -S WUXP:5000/scenario -A collect -M allrows_mix -T 0
-o metrics.xml -c config.xml -e elap_avg -d 5,5 -l 5 -i metrics.xml
-f a1.xml,a2.xml,a3.xml -v -s
```

Note Only users with sa_role and sso_role can run QPTune actions, except for compare, which may be run by any user.

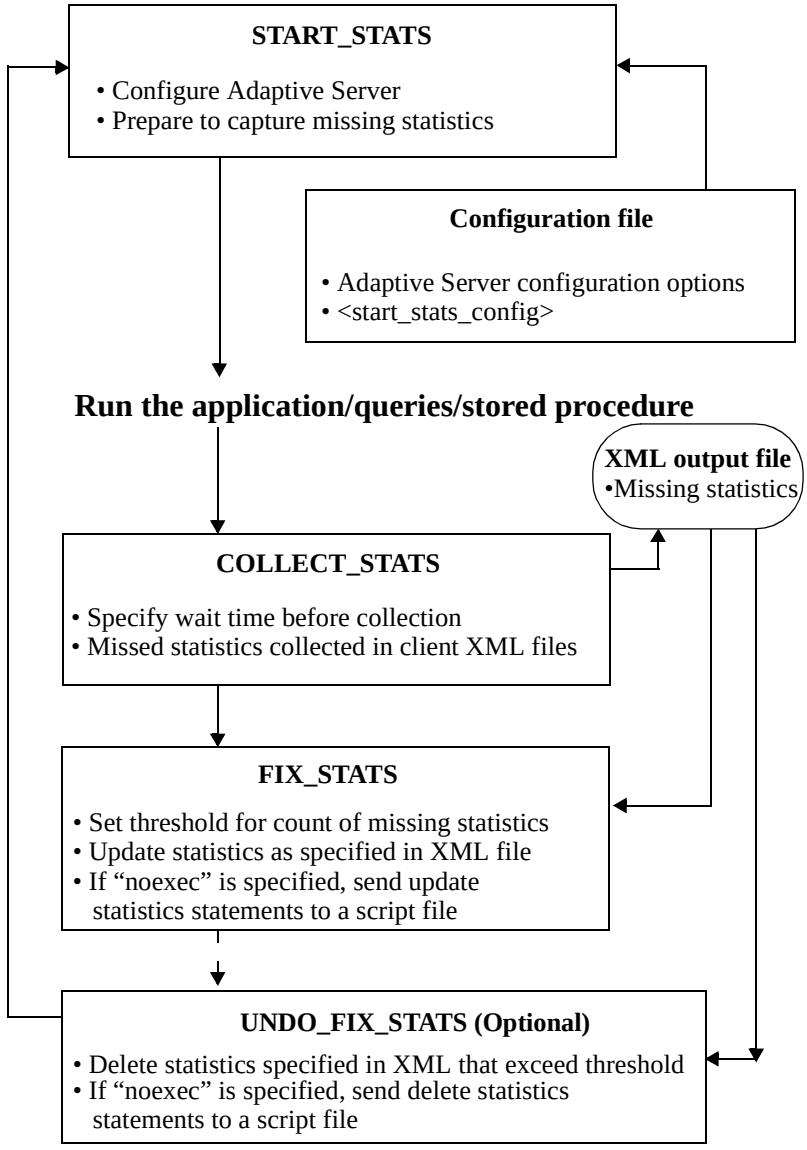
Using QPTune to fix missing statistics

Use QPTune to fix or update the missing statistics after you have upgraded a server. The main steps for using QPTune to fix missing statistics are:

- Start QPTune using the start_stats action.
- Run the application, queries, or stored procedure.
- Collect any missing statistics information into a specified XML file. See “Collecting statistics” on page 21.
- Use the fix_stats action to update statistics as specified in the above XML file. See “Fixing statistics” on page 23.
- (Optional) Undo the fix of missing statistics using the undo_fix_stats action. See “Using undo_fix_stats” on page 24.

The tuning cycle to fix missing statistics is shown here:

Figure 2-1: Tuning cycle to fix missing statistics



Starting QPTune to fix missing statistics

Start the utility with the start_stats action. For example:

```
QPTune -A start_stats -S my_host:4816/my_database
       -v

Executing : QPTune -U sa -P [unshown]
-S jdbc:sybase:Tds:my_host:4816/my_database
-A start_stats -M allrows_dss -T 0 -i null
-o metrics.xml -f null -c config.xml -l 5
-e elap_avg -d 5,5 -m 5 -n null -v
You are now connected to database: my_database
[INFO] Config: sp_configure 'capture missing
statistics', 1
[INFO] Config: sp_configure 'system table', 1
[INFO] Config: delete sysstatistics where formatid =110
```

You may also use the -c option to specify a configuration file. This extracts server-level configuration settings from the <start_stats> section of your configuration file. See “Configuration file” on page 32.

Collecting statistics

After preparing the system by running QPTune with the start_stats action, you may begin collecting the missing statistics with the collect_stats action. You can have QPTune either perform this action immediately, or after waiting for some period of time. This feature enables you to automate the start_stats and collect_stats steps.

collect_stats retrieves missing statistics information from the sysstatistics table for statistics that exceed a specified threshold for count of missing statistics. QPTune consolidates the missing statistics and determines a minimum set of statistics that must be updated.

The -m option indicates the threshold for count of missing statistics. When the statistics for a query have been missed as many times as the threshold value or more, they are collected and exported to an XML file. The default threshold count is 5.

The -o option indicates the output XML file that holds missing statistics. Use the output XML from collect_stats as input to the fix_stats and undo_fix_stats actions.

For example:

```
QPTune -A collect_stats -m 1 -o missingstats.xml -v
```

```
        -S my_host:4816/my_database

Executing : QPTune -U sa -P [unshown] -S
jdbc:sybase:Tds:my_host:4816/my_database -A
collect_stats -M allrows_dss -T 0 -i null -o
missingstats.xml -f null -c config.xml -l 5 -e elap_avg
-d 5,5 -m 1 -n null -v
You are now connected to database: my_database
Now collecting missing statistics information from
sysstatistics on "Fri Sep 26 10:08:06 PDT 2008".
<?xml version="1.0" encoding="UTF-8"?>
<server url="jdbc:sybase:Tds:my_host:4816/my_database"
file="missingstats.xml"
type="missing stats" datetime="Fri Sep 26 10:08:06 PDT
2008" >
<missingStat id="1">
<id>1068527809</id>
<stats>Y(y4,y2)</stats>
<count>2</count>
</missingStat>
<missingStat id="2">
<id>1068527809</id>
<stats>Y(y3)</stats>
<count>1</count>
</missingStat>
<missingStat id="3">
<id>1068527809</id>
<stats>Y(y2,y1)</stats>
<count>1</count>
</missingStat>
<missingStat id="4">
<id>1068527809</id>
<stats>Y(y1)</stats>
<count>1</count>
</missingStat>
</server>
The missing statistics information is written into XML
file: missingstats.xml
[INFO] End config: sp_configure 'enable metrics
capture', 0
[INFO] End config: sp_configure 'abstract plan dump', 0
[INFO] End config: sp_configure 'system table', 0
[INFO] End config: sp_configure 'capture missing
statistics', 0
Program has restored the data source for metrics
collection.
----- QPTune finished executing. ------
```

Fixing statistics

After collecting missing statistics information into an XML file, you can update the statistics that are equal to, or exceed, the threshold for count of missing statistics specified by the -m option. Use the fix_stats action to update statistics.

The -i option specifies the input XML file that contains all missing statistics.

You can generate a SQL script for updating statistics without executing the actual updates by using the -N option to indicate "noexec", and the -o option to indicate the output script file. The output file is created with all the generated update statistics statements but the statements are not executed. Generated scripts have a SQL file format. Using the -N option gives you the option of running the SQL script at a later time to optimize your resources.

For example:

```
QPTune -A fix_stats -m 1 -i missingstats.xml
        -v -S my_host:4816/my_database

Executing : QPTune -U sa -P [unshown] -S
jdbc:sybase:Tds:my_host:4816/my_database -A fix_stats -
M allrows_dss -T 0 -i missingstats.xml -o metrics.xml -
f null -c config.xml -l 5 -e elap_avg -d 5,5 -m 1 -n
null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:14:59 PDT 2008"
------------------------------------------------------
-----
Details of statements(s) fixed:
-------------------------------
Fixed statistics:[Update] Y(y4,y2)
[INFO] Fix Statement = update statistics Y(y4,y2)
Fixed statistics:[Update] Y(y3)
[INFO] Fix Statement = update statistics Y(y3)
Fixed statistics:[Update] Y(y2,y1)
[INFO] Fix Statement = update statistics Y(y2,y1)
Fixed statistics:[Update] Y(y1)
[INFO] Fix Statement = update statistics Y(y1)
----- QPTune finished executing. ------
```

For example:

```
QPTune -U sa -P -S my_host:5000/my_database
       -A fix_stats -m 5 -i missingstats.xml
        -N -o missingstats.sql
```

Using undo_fix_stats

To revert fixed missing statistics, use the undo_fix_stats action. undo_fix_stats deletes the statistics that are specified in an XML file whose missing counts are equal to, or exceed, the number specified by the -m option.

For example:

```
QPTune -A undo_fix_stats -m 1 -i missingstats.xml
        -v -S my_host:4816/my_database

Executing : QPTune -U sa -P [unshown] -S
jdbc:sybase:Tds:my_host:4816/my_database -A
undo_fix_stats -M allrows_dss -T 0 -i missingstats.xml
-o metrics.xml -f null -c config.xml -l 5 -e elap_avg -
d 5,5 -m 1 -n null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:20:23 PDT 2008"
------------------------------------------------------
-----
Details of statements(s) fixed:
-------------------------------
Fixed statistics:[Delete] Y(y4,y2)
[INFO] Fix Statement = delete statistics Y(y4,y2)
Fixed statistics:[Delete] Y(y3)
[INFO] Fix Statement = delete statistics Y(y3)
Fixed statistics:[Delete] Y(y2,y1)
[INFO] Fix Statement = delete statistics Y(y2,y1)
Fixed statistics:[Delete] Y(y1)
[INFO] Fix Statement = delete statistics Y(y1)
----- QPTune finished executing. ------
```

Using QPTune to tune queries or applications

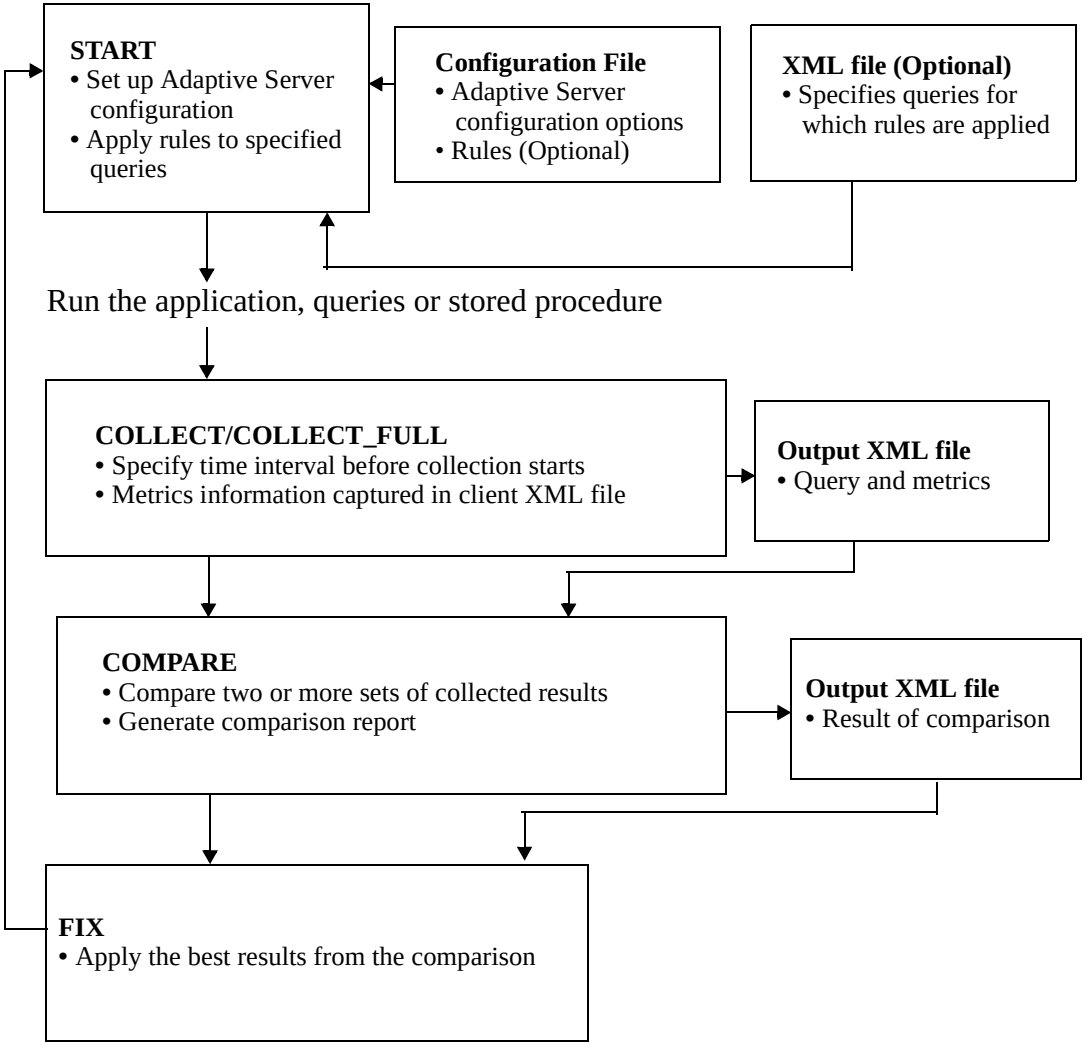
The main tasks for using QPTune for application or query tuning are:

- Start QPTune using either:
  - “Simple start” on page 27 if you are applying standard optimization goal settings.
  - “Custom start” on page 27 if you are applying special/custom rules to specified queries.
- Run the application, queries, or stored procedure you are tuning.

- Collect the metrics into specified XML files. See “Collecting metrics” on page 28.
- Compare the sets of metrics you have collected for different optimization goals. This step uses the XML file from the above step as input, and generates a performance comparison report. See “Comparing metrics” on page 29.
- Apply the best results from the comparison to each of the specified queries on the target server. See “Applying the best results” on page 30.

QPTune’s tuning cycle for applications or queries is shown here:

Figure 2-2: QPTune cycle for application or query tuning



Starting QPTune to tune queries or applications

The start action of QPTune prepares the server with correct server-level configuration settings. If a configuration file is used, the settings are extracted from the <start> section of the configuration file. The <end> section of the configuration file specifies the settings that enable the system to revert to its original state at the end of QPTune collect action. See "Configuration file" on page 32 for more information on the configuration file.

Simple start

If you are applying standard optimization goal settings, start QPTune using:

```
QPTune -S host:port/database -A start
        [-M {allrows_oltp, allrows_dss, allrows_mix}]
```

Use the -M option to invoke one of the pre-programmed modes that correspond to the three optimization goals in Adaptive Server:

- allrows_mix
- allrows_oltp
- allrows_dss (default)

Custom start

If you are applying custom rules to specified queries, use:

```
QPTune -S host:port/database -A start -M custom_1
        -i input.xml -l 3 [-v]
```

Use the -M option to indicate a custom mode. A custom mode is a group of special rules that are specified in the configuration file under the <mode> section. Rules are Adaptive Server 15.0 optimization criteria that are applicable at the query level using abstract query plans.

The example above uses a custom mode called *custom_1* which may be a combination of rules such as:

- use optgoal allrows_mix
- use merge_join off
- use opttimeoutlimit 15

Use the -i option to indicate an input XML file which has been generated by QPTune while applying a standard goal setting during the collect phase. A number of collected metrics files may be compared to generate a file with the best goal settings. The input file contains SQL text for the queries.

Use the -l option along with the -i option to indicate the number of queries that should be applied with these special rules. The queries are counted from the start of the file. The default value of the -l option is 0, which implies that all queries in the input file are applied.

Collecting metrics

After starting the system, run your applications and collect metrics into an XML file. Use the -o option to specify the output metrics file. The -v option provides a verbose output. The -M option indicates custom or standard modes.

You can collect metrics either:

- Immediately, using the -T 0 option, or
- After *t* minutes, using the -T *t* option.

For example, the command below writes XML into a file named *a2.xml*. The custom mode is depicted within the <bestmode> tags.

```
QPTune -S host:port/database -A collect -T 0
        -o a2.xml -v
```

```
Program has configured the data source for metrics collection.
Now collecting information from sysquerymetrics on "Tue Feb 19 22:16:04 PST
2008".
<?xml version="1.0" encoding="UTF-8"?>
 <server url="jdbc:sybase:Tds:SHANGHI:5000" type="ASE" mode="custom_1"
datetime="Tue Feb 19 22:16:04 PST 2008">
<query id="1">
<qtext> select count(T.title_id) from authors A, titleauthor T
where A.au_id = T.au_id </qtext>
<elap_avg>300</elap_avg>
<bestmode> custom_1
</bestmode>
</query>
</server>
```

Note You can use the output XML file from the collect operation as input to compare, fix, or start operations.

Comparing metrics

Once metrics are collected, you can compare different XML files to get the best query optimization goal or criteria for each of the queries. For example:

```
QPTune -A compare -f a1.xml,a2.xml[,a3.xml..] -d 51,10
     -o best.xml -S my_host:5000/my_database
```

The -f option specifies a list of two or more collected metrics sample files separated by commas. Use quotes to encapsulate the file name if it contains any spaces.

The -d option indicates a threshold percentage and absolute value. A performance improvement beyond the threshold percentage and absolute value is considered "outstanding" during the fix operation. The optimization goal/criteria for those outstanding queries is applied to the server as a plan fix.

The default for the threshold percentage and absolute value pair is "5,5". If only percentage is specified, the absolute value defaults to 0. Percentage values are between 0 and 100; an absolute value can be any number greater than 0.

The -o option specifies the result of the comparison in a file. The file holds the best setting for all the queries being analyzed.

The -s option enables sorting the files from largest to smallest. The file with the largest set of queries is used as the basis for comparison.

The following example shows the result of a compare operation:

```
Compare all the files: | a1.xml, a2.xml|
Report generated on "Tue Aug 19 21:13:04 PST 2008"
--------------------------------------------------------------------------
File #1: [name= a1.xml  :  mode=allrows_mix]
File #2: [name= a2.xml  :  mode=custom_1]
Query count in File #1 : [mode=allrows_mix]         6
Query count in File #2 : [mode=custom_1]            7
==========================================================================
Query count improved in File #2: [mode=allrows_mix] 3

Total performance improved [from 422 to 129]: 69 %

Following queries run better in File #2:
[mode=allrows_mix]
--------------------------------------------------------------------------
Group 1: improved by no more than 25% [0 queries]
Group 2: improved by 25% to 50% [1 queries]
Query: select count(T.title_id) from authors A, titleauthors T where A.au_id =
T.au_id
Average elapsed time (ms): File #1=100  File #2=50   Improvement=50.0%
```

```
Outstanding=No
Group 3: improved by 50% to 75% [0 queries]
Group 4: improved by 75% to 100% [2 queries]
Query: select count(*) from titlles T, titleauthors TA where T.title_id =
TA.title_id
Average elapsed time (ms): File #1=34  File #2=7       Improvement=79.0%
Outstanding=Yes
Query: select au_lname, au_fname from authors where state in ("CA", "AZ")
Average elapsed time (ms): File #1=9   File #2=0       Improvement=100.0%
Outstanding=No
```

The above example shows a comparison between two XML metrics files: *a1.xml* has six queries, and *a2.xml* has seven queries. Comparisons can only be made between the queries that are common to both files. There are three queries that ran faster in *a2.xml*. The improvements are categorized into four groups:

- Group 1 – between 0 and 25%
- Group 2 – between 25% and 50%
- Group 3 – between 50%and 75%
- Group 4 – between 75% and 100%

There is one query between 25 and 50% and two queries between 75% and 100%. The queries in Group 2 are marked as “Outstanding=No” which means that based on the threshold of 51%, this query will not be fixed.

While comparing more than two files, QPTune updates the first file with the best from both files, then compares the new file with the third file, and so on.

Applying the best results

After getting the results for all queries being analyzed, use the fix action to apply the best settings to the queries in the database system.

For example:

```
QPTune  -S host:port/database -A fix -i best.xml
         -v -g
```

The -i option specifies the queries and their best plans resulting from the comparison.

The -g option, when used with the fix action, applies the default goal. The default goal is the best optgoal setting that most queries used as the best plan using QPTune's fix action. This option only generates plans for queries that do not currently use the server's default optimization goal.

The example fix action above produces this output:

```
Query Plan(s) fixed on "Wed Sep 17 17:44:09 PDT 2008"
------------------------------------------------------------
Fixed 2 queries using mode "custom_1" with following optimizer settings": '(use
optgoal allrows_mix) (use merge_join off) (use opttimeoutlimit 15)'

Fixed 4 query using mode "allrows_mix"

Apply "sp_configure optimization_goal, 0, allrows_mix" as the default optgoal
Details of statement(s) fixed:
---------------------------------------
Query: 'select count(T.title_id) from authors A, titleauthor T where A.au_id =
T.au_id '
Fixed using: 'custom_1'
[INFO] Fix Statement = create plan 'select count(T.title_id) from authors A,
titleauthor T where A.au_id = T.au_id' '(use optgoal allrows_mix) (use
merge_join off) (use opttimeoutlimit 15)'
Query: 'select * from titleauthors where au_id  > 20 and title_id < 100'
Fixed using: 'custom_1'
[INFO] Fix Statement = create plan 'select * from titleauthors where au_id  >
20 and title_id < 100' '(use optgoal allrows_mix) (use merge_join off) (use
opttimeoutlimit 15)'
```

QPTune then creates an optimized query plan which is saved in the sysqueryplans system table in the current database. When a query with matching SQL is encountered, this optimized plan is used. Incoming SQL and the SQL of the persistent plan are said to match when a checksum type of hash on the two SQL statements matches. If literal parameterization is enabled explicitly, the two statements may differ only in the static values of search arguments such as:

```
where CustomerID = "12345"
```

In this case, the value "12345" is replaced by a placeholder variable, so the hash value is the same, regardless of the search value.

If the application changes the SQL in any manner, such as adding a new predicate, there is no longer a match to a persistent plan and the optimizer creates a query plan according to the current configuration and available statistics.

Configuration file

You can define custom modes in a configuration file. The QPTune installation includes a standard configuration file that contain some custom modes. The custom mode "_basic_" is reserved for "basic optimization".

The configuration file for QPTune must include <start>, <start_stats>, <fix> and <end> sections. The <mode> section is optional.

The <start> section indicates the configuration settings for Adaptive Server before metrics are collected. For example:

```
 <start>
<!-- Recommended server settings -->
<start_config>sp_configure 'enable metrics capture', 1</start_config>
<start_config>sp_configure 'abstract plan dump', 1</start_config>

<!-- Clean up sysqueryplans & sysquerymetrics tables -->

<start_config>sp_configure 'system table', 1</start_config>
<start_config>sp_metrics 'flush'</start_config>
<start_config>delete sysqueryplans where gid=1 or gid=2</start_config>

<!-- Optional settings that users can change or remove -->

<!-- <start_config>sp_configure 'enable literal autoparam', 1</start_config> ->
<!-- <start_config>sp_configure 'metrics elap max', 0</start_config> -->

<!-- Hint: sp_add_resource_limit can be added to limit resource usage  -->
<!-- Specify a query plan group name to save all existing plans from ap_stdin -->
<!-- Existing plans from ap_stdout will be saved to the corresponding group name
added with '_out'. -->

<save_plans_pre_start>pre_start_qpgroup</save_plans_pre_start>
</start>
```

The <end> section corresponds to the <start> section and includes the configurations setting to be applied after metrics are collected. For example:

```
<end>
<end_config>sp_configure 'enable metrics capture', 0</end_config>
<end_config>sp_configure 'abstract plan dump', 0</end_config>
<end_config>sp_configure 'system table', 0</end_config>
<end_config>sp_configure 'capture missing statistics', 0</end_config>

<!-- <end_config>sp_configure 'enable literal autoparam', 0</end_config> -->
<!-- <end_config>sp_configure 'metrics elap max', 0</end_config> -->
</end>
```

The <start_stats> section includes statistics settings. For example:

```
<start_stats>
<!-- Recommended server settings -->
<start_stats_config>sp_configure 'capture missing
statistics',1</start_stats_config>
<!-- Reset counter of missing statistics -->
<start_stats_config>
sp_configure 'system table',1
</start_stats_config>
<start_stats_config>
delete sysstatistics where formatid=110
</start_stats_config>
</start_stats>
```

The <fix_stats>section includes:

```
<!-- The following set of configurations apply at "-A
fix" -->
<fix>
<!-- Recommended server settings -->
<fix_config>sp_configure 'abstract plan load',1</fix_config>
<!-- Clean up sysqueryplans & sysquerymetrics tables -->
<fix_config>sp_configure 'system table', 1</fix_config>
<fix_config>sp_metrics 'flush'</fix_config>
<fix_config>delete sysqueryplans where gid=1 orgid=2</fix_config>
<!-- Optional settings that users can change or remove -->
<fix_config>sp_configure 'enable metrics capture',1</fix_config>
<!-- <fix_config>sp_configure 'enable literal autoparam',1</fix_config> -->
<!-- <fix_config>sp_configure 'metrics elap max',0</fix_config>-->
<!-- Specify a query plan group name to save all existing plans from ap_stdin -->
<!-- Existing plans from ap_stdout will be saved to the corresponding group name
added with '_out'. -->
<save_plans_pre_fix>pre_fix_qpgroup</save_plans_pre_fix>
</fix>
```

The optional <mode> section allows users to specify custom optimization settings to one or more queries specified through another input file. The -M option of the start and collect actions specifies the mode setting. When the -M option specifies anything other than a standard optimization goal setting, QPTune treats the mode as customized, and retrieves the optimization goal and rules settings, for the indicated name, from the <mode> section of the configuration file. QPTune then applies the custom settings to the list of specified queries.

Examples

❖ **Fixing missing statistics using QPTune**

1 Run QPTune with start_stats to prepare the server to collect missing statistics:

```
QPTune -A start_stats -v
        -S my_host:4816/my_database
```

Sample output:

```
Executing : QPTune -U sa -P [unshown]
-S jdbc:sybase:Tds:my_host:4816/my_database
-A start_stats -M allrows_dss -T 0 -i null -o metrics.xml
-f null -c config.xml -l 5 -e elap_avg -d 5,5
-m 5 -n null –v
You are now connected to database: my_database
[INFO] Config: sp_configure 'capture missing statistics', 1
[INFO] Config: sp_configure 'system table', 1
[INFO] Config: delete sysstatistics where formatid =110
```

2 Run the client application, stored procedure, or query.

3 Run QPTune with collect_stats action to collect statistics that exceed the threshold for count of missing statistics. You may let the utility wait for some period of time (specified by the -T option) before collecting the missing statistics information.

```
QPTune -A collect_stats -m 1 -o missingstats.xml
        -v -S my_host:4816/my_database
```

Sample output:

```
Executing : QPTune -U sa -P [unshown]
-S jdbc:sybase:Tds:my_host:4816/my_database
-A collect_stats -M allrows_dss -T 0 -i null -o missingstats.xml -f null
-c config.xml -l 5 -e elap_avg -d 5,5 -m 1 -n null -v
You are now connected to database: my_database
Now collecting missing statistics information from sysstatistics on "Fri
Sep 26 10:08:06 PDT 2008".
QPTune Utility
<?xml version="1.0" encoding="UTF-8"?><server
url="jdbc:sybase:Tds:my_host:4816/my_database"
file="missingstats.xml"
type="missing stats" datetime="Fri Sep 26 10:08:06 PDT 2008" >
  <missingStat id="1">
     <id>1068527809</id>
     <stats>Y(y4,y2)</stats>
```

```
      <count>2</count>
   </missingStat>
   <missingStat id="2">
      <id>1068527809</id>
      <stats>Y(y3)</stats>
      <count>1</count>
   </missingStat>
   <missingStat id="3">
      <id>1068527809</id>
      <stats>Y(y2,y1)</stats>
      <count>1</count>
   </missingStat>
   <missingStat id="4">
      <id>1068527809</id>
      <stats>Y(y1)</stats>
      <count>1</count>
   </missingStat>
</server>
The missing statistics information is written into XML file:
missingstats.xml
[INFO] End config: sp_configure 'enable metrics capture', 0
[INFO] End config: sp_configure 'abstract plan dump', 0
[INFO] End config: sp_configure 'system table', 0
[INFO] End config: sp_configure 'capture missing statistics', 0
Program has restored the data source for metrics collection
----- QPTune finished executing. ------
```

4 Update statistics that have exceeded or equalled the threshold for count of missing statistics, specified by the -m option. To fix missing statistics that are specified in the input file *missingstats.xml*, use:

```
QPTune -U sa -P -A fix_stats -m 1 -i missingstats.xml
       -v -S my_host:4816/my_database
```

Sample output:

```
Executing : QPTune -U sa -P
-S jdbc:sybase:Tds:my_host:4816/my_database -A fix_stats
-M allrows_dss -T 0 -i missingstats.1 xml -o metrics.xml -f null
-c config.xml -l 5 -e elap_avg -d 5,5 -m 1 -n null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:14:59 PDT 2008"
------------------------------------------------------------
Details of statements(s) fixed:
------------------------------
Fixed statistics:[Update] Y(y4,y2)
[INFO] Fix Statement = update statistics Y(y4,y2)
Fixed statistics:[Update] Y(y3)
```

```
[INFO] Fix Statement = update statistics Y(y3)
Fixed statistics:[Update] Y(y2,y1)
[INFO] Fix Statement = update statistics Y(y2,y1)
Fixed statistics:[Update] Y(y1)
[INFO] Fix Statement = update statistics Y(y1)
----- QPTune finished executing. ------
```

Note If the fix_stats action is used with the -N option, QPTune does not execute the statements to fix missing statistics, but instead sends them to an output file specified by -o *output_file*.

5 (Optional) The undo_fix_stats command deletes the statistics specified in the -i XML file. The statistics deleted are those that have missing counts exceeding or equal to a number specified by -m. To undo the fix of missing statistics in the input file *missingstats.xml* use:

```
QPTune -U sa -P -A undo_fix_stats -m 1
       -i missingstats.xml -v
       -S my_host:4816/my_database
```

Sample output:

```
Executing : QPTune -U sa -P [unshown]
-S jdbc:sybase:Tds:my_host:4816/my_database
-A undo_fix_stats -M allrows_dss -T 0
-i missingstats.xml -o metrics.xml -f null
-c config.xml -l 5 -e elap_avg -d 5,5 -m 1
-n null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:20:23 PDT 2008"
------------------------------------------------------------
Details of statements(s) fixed:
------------------------------
Fixed statistics:[Delete] Y(y4,y2)
[INFO] Fix Statement = delete statistics Y(y4,y2)
Fixed statistics:[Delete] Y(y3)
[INFO] Fix Statement = delete statistics Y(y3)
QPTune Utility
Fixed statistics:[1 Delete] Y(y2,y1)
[INFO] Fix Statement = delete statistics Y(y2,y1)
Fixed statistics:[Delete] Y(y1)
[INFO] Fix Statement = delete statistics Y(y1)
----- QPTune finished executing. ------
```

❖ **Optimizing an application using QPTune**

1 Run QPTune with start, specifying one of: allrows_oltp, allrows_mix, or allrows_dss:

```
QPTune -U sa -P -S my_host:11030/my_database
        -A start -M allrows_mix -v
```

In this example, Adaptive Server runs on a machine called "my_host" with a port number 11030 and a database called my_database.

Sample output:

```
Executing : QPTune -Usa -P [unshown] –S
jdbc:sybase:Tds:my_host:11030/my_database
-A start -M allrows_mix -T 0 -i null -o metrics.xml -f null -c config.xml
-l 5
-e elap_avg -d 5,5 -n null -v
You are now connected to database: my_database
[INFO] Config: sp_configure 'enable metrics capture', 1
[INFO] Config: sp_configure 'abstract plan dump', 1
[INFO] Config: sp_configure 'system table', 1
[INFO] Config: sp_metrics 'flush'
[INFO] Config: delete sysqueryplans
Apply "sp_configure optimization_goal, 0, allrows_mix" to the data
source.
Program has configured the data source for metrics collection.
```

2 Run the client application, stored procedure, or query. The client application may be a GUI-based or Web-based program, a set of stored procedures, or a batch of SQL queries in a script. For example:

```
isql -Usa -P < sp_telco.sql > sp_telco_allrows_mix.out
```

3 Run QPTune with collect to collect metrics on each of the queries in the application. The metrics are collected in a file called *sp_telco_allrows_mix.xml*.

```
QPTune -U sa -P -S my_host:11030/my_database -A collect
     -M allrows_mix -o sp_telco_allrows_mix.xml -v
```

Repeat steps 1 – 3 for each of other optimization goals or custom modes. For example, to use allrows_dss, run:

```
QPTune -U sa -P -S my_host:11030/my_database -A start
     -M allrows_dss

isql -Usa -P < sp_telco.sql > sp_telco_allrows_dss.out
QPTune -U sa -P -S my_host:11030/my_database -A collect
     -M allrows_dss -o sp_telco_allrows_dss.xml
```

Sample output for mode allrows_mix:

```
Executing : QPTune -U sa -P [not shown]
     -S jdbc:sybase:Tds:my_host:11030/my_database –A collect
     -M allrows_mix -T 0 -i null -o sp_telco_allrows_mix.xml -f null
     -c config.xml-l 5 -e elap_avg -d 5,5 -n null -v
You are now connected to database: my_database
Now collecting information from sysquery tables on "Tue Aug 26 22:00:49
PDT 2008".
Metrics are flushed.
<?xml version="1.0" encoding="UTF-8"?>
<server url="jdbc:sybase:Tds:my_host:11030/my_database"
file="sp_telco_allrows_mix.xml" mode="allrows_mix"
datetime="Tue Aug 26 22:00:49 PDT 2008" >
<query id="1">
<qtext>SELECT service_key , year , fiscal_period , count(*)
FROM telco_facts T , month M , status S
WHERE T.month_key=M.month_key AND S.status_key = T.status_key
AND call_waiting_status="Dropped"
GROUP BY year , fiscal_period , service_key
ORDER BY year , fiscal_period , service_key </qtext>
<hashkey>323626785</hashkey>
<id>1568005586</id>
<elap_avg>27408</elap_avg>
<bestmode>allrows_mix</bestmode>
</query>
<query id="2">
<qtext>SELECT customer_last_name , customer_first_name FROM
residential_customer R , telco_facts T , service S , month M
WHERE M.month_text = 'February ' AND M.year = 1998
AND S.isdn_flag = 'Y' AND M.month_key = T.month_key
AND S.service_key = T.service_key
AND R.customer_key = T.customer_key -- end comment i
</qtext>
<hashkey>727793461</hashkey>


<id>1552005529</id>
<elap_avg>3355</elap_avg>
<bestmode>allrows_mix</bestmode>
</query>
. . . . . .
<query id="10">
<qtext>SELECT month_key , service_key , count(*)
FROM telco_facts WHERE month_key = 1
GROUP BY month_key , service_key
</qtext>
```

```
<hashkey>1561133104</hashkey>
<id>1680005985</id>
<elap_avg>58</elap_avg>
<bestmode>allrows_mix</bestmode>
</query>
</server>
The metrics information is written into
XML file: sp_telco_allrows_mix.xml
[INFO] End config: sp_configure 'enable metrics capture', 0
[INFO] End config: sp_configure 'abstract plan dump', 0
[INFO] End config: sp_configure 'system table', 0
Program has restored the data source for metrics collection.
----- QPTune finished executing. ------
```

4 Compare metrics collected from all the runs, with the best metrics for each query, in a file called *best.xml*. You can define a new mode, called "new_mode" for this metric.

```
QPTune -U sa -P -S my_host:11030/my_database -v -A compare -M new_mode
     -f sp_telco_allrows_dss.xml,
     sp_telco_allrows_mix.xml,sp_telco_allrows_oltp -o best.xml
```

Sample output:

```
Executing: QPTune -U sa -P [unshown]

     -S jdbc:sybase:Tds:my_host:11030/my_database
     -A compare -M new_mode -T 0 -I null -o best.xml
     –f sp_telco_allrows_mix.xml, sp_telco_allrows_dss.xml,
     sp_telco_allrows_oltp.xml
     -c config.xml -l 5 -e elap_avg -d 5,5 -n null –v
Compare all the files:
sp_telco_allrows_mix.xml,sp_telco_allrows_dss.xml,sp_telco_allrows_oltp
.xml
Report generated on "Wed Aug 27 16:29:01 PDT 2008"
------------------------------------------------------
Sorted List By File Size (Desc.) =
sp_telco_allrows_mix.xml, sp_telco_allrows_dss.xml,
sp_telco_allrows_oltp.xml
File #1 : [name=sp_telco_allrows_mix.xml : mode=allrows_mix]
File #2 : [name=sp_telco_allrows_dss.xml : mode=allrows_dss]
Query count in File #1 [mode=allrows_mix]: 14
Query count in File #2 [mode=allrows_dss]: 12
================================================
Query count improved in File #2[mode=allrows_dss]: 7
Total performance improved [from 37234 to 7781]: 79 %
Following queries run better in File #2 [mode=allrows_dss]:
------------------------------------------------
```

```
Group 1: improved by no more than 25% [2 queries]
Query: SELECT state, count(*) FROM telco_facts T, service S,
residential_customer C, month M
WHERE T.service_key = S.service_key
AND T.customer_key = C.customer_key AND T.month_key = M.month_key
AND call_waiting_flag = 'Y' AND caller_id_flag = 'Y'
AND voice_mail_flag = 'N' AND state in ('NY', 'NJ', 'PA')
AND fiscal_period = 'Q1'
GROUP BY state
Average elapsed time(ms): File #1=837 File #2=803 Improvement=4.0%
Outstanding=No
Query: SELECT fiscal_period, T.service_key, sum(local_call_minutes),
sum(local_call_count) , count(*)
FROM telco_facts T ,residential_customer C, service S , month M
WHERE T.customer_key = C.customer_key
AND T.service_key = S.service_key AND T.month_key = M.month_key
AND fiscal_period = 'Q4' AND T.service_key in (02, 03)
AND state = 'CA'
GROUP BY fiscal_period , T.service_key
Average elapsed time(ms): File #1=832 File #2=635 Improvement=23.0%
Outstanding=Yes
-------------------------------------------------
Group 2: improved by 25% to 50% [2 queries]Group 3: improved by 50% to
75% [0 queries]
. . . . . .
Group 4: improved by 75% to 100% [3 queries]
Query: SELECT service_key , year , fiscal_period , count(*)
FROM telco_facts T , month M , status S
WHERE T.month_key=M.month_key AND S.status_key = T.status_key
AND call_waiting_status="Dropped" GROUP BY year, fiscal_period,
service_key
ORDER BY year , fiscal_period , service_key -- end comment--
Average elapsed time(ms): File #1=27408 File #2=2126 Improvement=92.0%
Outstanding=Yes
. . . . . .
File #3 : [name=sp_telco_allrows_oltp.xml : mode=allrows_oltp]
Query count in File #3[mode=allrows_oltp]: 13
=================================================
Query count improved in File #3[mode=allrows_oltp]: 4
Total performance improved [from 7781 to 6523]: 16 %
Following queries run better in File #3:

-------------------------------------------------
Group 1: improved by no more than 25% [2 queries]
Query: SELECT fiscal_period , count(*) , sum(local_call_minutes)
FROM residential_customer R , telco_facts T , status S , month M
WHERE S.call_waiting_status=@status AND state =
```

```
. . . . . .
```

5 Fix the query plans in your application by using the best plan from the comparison:

```
QPTune -U sa -P -S my_host:11030/my_database -g
        -A fix -i best.xml
```

Sample output:

```
Executing : QPTune -U sa -P [unshown]
-S jdbc:sybase:Tds:my_host:11030/my_database
-A fix -M allrows_dss -T 0 -i best.xml
-o metrics.xml -f null -c config.xml -l 5
-e elap_avg -d 5,5 -n null -v
You are now connected to database: my_database
[INFO] Config: sp_configure 'abstract plan load', 1
[INFO] Config: sp_configure 'system table', 1
[INFO] Config: sp_metrics 'flush'
[INFO] Config: delete sysqueryplans
[INFO] Config: sp_configure 'enable metrics capture', 1
You are now connected to database: my_database
Query plan(s) fixed on "Wed Aug 27 17:05:46 PDT 2008"
------------------------------------------------------------
Fixed 3 queries using mode "allrows_oltp"
Fixed 3 queries using mode "allrows_dss"
Fixed 8 queries using mode "allrows_mix"
Apply "sp_configure optimization_goal, 0, allrows_mix" as the
default optgoal.
Details of statements(s) fixed:
------------------------------
Query: SELECT service_key , year , fiscal_period , count(*) -- comment 1
it''s a comment. whatever "statments"
/* comment 3 */ FROM telco_facts T , month M , status S
WHERE T.month_key=M.month_key AND S.status_key = T.status_key
AND call_waiting_status="Dropped" GROUP BY year, fiscal_period,
service_key
ORDER BY year , fiscal_period , service_key -- end comment
-- *** Query #9 ***
. . . . . .
```

❖ **Using QPTune custom modes**

1 You may run select queries using your own custom modes defined in a configuration file. QPTune includes some custom modes like "_basic_," which represents basic optimization of Adaptive Server 12.5. For example, the default configuration file *config.xml* contains custom mode "custom1" which allows an optimization goal of allrows_oltp, together with the rule merge_join_off:

```
<!-- "default" custom mode -->
<mode name="default">
<optgoal>use optgoal allrows_mix</optgoal>
<rule>use merge_join off</rule>
<rule>use opttimeoutlimit 15</rule>
</mode>
<!-- "_basic_" mode is a reserved system mode. -->
<mode name="_basic_">
</mode>
<mode name="custom1">
<optgoal>use optgoal allrows_oltp</optgoal>
<rule>use merge_join off</rule>
</mode>
```

2 This example shows how to use the "_basic_" custom mode:

```
QPTune –A start –M _basic_ –Usa –P –S my_host:11030/my_database
       –i best.xml -l 0 -v
isql -Usa -P < sp_telco_2.sql > sp_telco_basic.out
QPTune -A collect -M _basic_ -Usa -P -S my_host:11030/my_database
       -o sp_telco_basic.xml –v
QPTune -A compare -M best -Usa -P -Smy_host:11030/my_database -v
       -f sp_telco_basic.xml,best.xml -o best_basic.xml –d 1,0
```

Sample output:

```
Report generated on "Fri Aug 29 13:29:17 EDT 2008"
------------------------------------------------------
{INFO]Sorted List By File Size (Desc.)=sp_telco_basic.xml,best.xml
File #1 : [name=sp_telco_basic.xml : mode=_basic_]
File #2 : [name=best.xml : mode=best]
Query count in File #1: 14
Query count in File #2: 14
================================================
Query count improved in File #2: 7
Total performance improved [from 2441 to 1529]: 37 %
Following queries run better in File #2:
------------------------------------------------
Group 1: improved by no more than 25% [4 queries]
Query: SELECT customer_last_name , customer_first_name
```

```
FROM residential_customer R , telco_facts T , service S , month M
WHERE M.month_text = 'February ' AND M.year = 1998
AND S.isdn_flag = 'Y' AND M.month_key = T.month_key
AND S.service_key = T.service_key AND R.customer_key = T.customer_key
Average elapsed time(ms): File #1=393 File #2=306 Improvement=22.0%
Outstanding=Yes
. . . . . .
```

Upgrade issues

QPTune helps you get the best performance when upgrading to Adaptive Server 15.0 . If there are queries that do not perform as well as pre 15.0 versions of the server, QPTune allows Adaptive Server Enterprise to generate version 12.5.4-like query plans. If these plans run faster than the corresponding version 15.0 query plans, QPTune retains and uses these plans for all subsequent query execution.

❖ **Using QPTune while migrating to Adaptive Server 15.0**

1 Depending on the application, get metrics information for any or all of the three Adaptive Server 15.0's pre-programmed modes (“mix,” “dss,” “oltp”):

```
QPTune -A start -M allrows_oltp
        -S my_host:5000/my_database
```

<Run your query, application, or stored procedure>

```
QPTune -A collect -M allrows_oltp -T 0 -o oltp.xml
        -S my_host:5000/my_database

QPTune -A start -M allrows_mix
        -S my_host:5000/my_database
```

<Run your query, application, or stored procedure>

```
QPTune -A collect -M allrows_mix -T 0 -o mix.xml
        -S my_host:5000/my_database

QPTune -A start -M allrows_dss
        -S my_host:5000/my_database
```

<Run your query, application, or stored procedure>

```
QPTune -A collect -M allrows_dss -T 0 -o dss.xml
        -S my_host:5000/my_database
```

2 Get metrics information with optimization similar to version 12.5.4 in Adaptive Server 15.0:

```
QPTune -A start -M _basic_ -i oltp.xml -l 10
        -S my_host:5000/my_database
```

<Run your query, application, or stored procedure>

```
QPTune -A collect -M _basic_ -T 0 -o basic.xml
        -S my_host:5000/my_database
```

3 Compare the metrics information:

```
QPTune -A compare -d 10 -o best.xml
        -f basic.xml,oltp.xml,mix.xml,dss.xml
        -S my_host:5000/my_database
```

4 Fix query plans with the best out of the comparison:

```
QPTune -A fix -i best.xml
        -S my_host:5000/my_database
```

5 (Optional) Verify the performance improvement after the plan fixup, re-run the application, and collect the metrics information:

```
QPTune -A collect -T 0 -o new_best.xml
        -S my_host:5000/my_database
```

Performing a compare of *new_best.xml* with any of the other XML output files should establish that *new_best.xml* gives the best results.

Localization

The QPTune command line utility has been localized so its messages can display in these 9 languages other than English: Chinese, French, German, Japanese, Korean, Polish, Portuguese, Spanish, and Thai. The language properties files are packaged in the *qptune.jar* file. QPTune sets the display according to the language set on the system's default locale.

QPTune GUI

The QPTune GUI is a set of Java libraries that are used by the Adaptive Server plug-in.

Environment and system requirements

To access QPTune functionality, you must be using Adaptive Server version 15.0.3 ESD #1 or later.

The QPTune executable and libraries are installed in:

(Unix)

$SYBASE/$SYBASE_ASE

$SYBASE/$SYBASE_ASE/lib

(Windows)

%SYBASE%\%SYBASE_ASE%

%SYBASE%\%SYBASE_ASE%\lib

To run the QPTune GUI, these files must be present in your installation:

- (UNIX)
  - *$SYBASE/$SYBASE_ASE/qptune/config.xml*
  - *$SYBASE/$SYBASE_ASE/lib/qptune.jar*
  - *$SYBASE/$SYBASE_ASE/qptune/lib/xercesImp.jar*
  - *$SYBASE/$SYBASE_ASE/qptune/lib/xml-apis.jar*
  - *$SYBASE/jConnect-6_0/classes/jconn3.jar*
  - *$SYBASE_JRE6/bin/java*
- (Windows)
  - *%SYBASE%\%SYBASE_ASE%\qptune\config.xml*
  - *%SYBASE%\%SYBASE_ASE%\lib\qptune.jar*
  - *%SYBASE%\%SYBASE_ASE%\qptune\lib\xercesImp.jar*
  - *%SYBASE%\%SYBASE_ASE%\qptune\lib\xml-apis.jar*
  - *%SYBASE%\jConnect-6_0\classes\jconn3.jar*
  - *%SYBASE_JRE6%\bin\java*

Set the following environment variables:

- SYBASE_JRE6 – to the Java runtime installation.
- SYBASE – to the latest Sybase installation on your machine.

- SYBASE_ASE – to the Adaptive Server component(directory) of the installation on your machine.

Starting the QPTune GUI

The QPTune GUI uses the ASE plug-in in Sybase Central. You must have installed Sybase Central in order to access the QPTune GUI. For more information on the ASE plug-in in Sybase Central, see the *System Administration Guide, Volume 1*.

Start Sybase Central, and configure your servers using the ASE plug-in. For any server that you can access with the plug-in, the QPTune GUI allows you to:

- Fix missing statistics in Adaptive Server:

  After a server upgrade, and before tuning the server, use QPTune to update missing statistics on the server.

- Tune tasks in Adaptive Server:

  Use this feature to define a tuning task that QPTune can execute and analyze in a report. You can further apply the fixes to the server.

Fixing missing statistics

Use the QPTune GUI to fix or update the missing statistics after you have upgraded a server. To access QPTune's "Fix missing statistics" feature, right-click a server name and select one of the two available menu options for fixing missing statistics:

- Fix Missing Statistics: When you select this option, QPTune brings up the Fix Missing Statistics Wizard. QPTune collects the information about missing statistics into an XML file, and then uses the file to fix the missing statistics. Specify the name of the XML file and the target database, and click Next for the Options page.

  On the Options page, specify a threshold for the count of missing statistics. The default threshold count is 5. See "Collecting statistics" on page 21 for more information on threshold for count of missing statistics.

  Additionally, you may opt to only send the update statistics statements to a script file. To do this, type in the filename where you would like to save the statements, and click Finish to save the tuning task without execution.

If you click Execute, QPTune goes through the steps to fix the missing statistics. QPTune then displays a Summary page of commands that the wizard issues to it.

- Undo Missing Statistics Fix: When you select this option, QPTune brings up the Undo Missing Statistics Fix Wizard. Using statistics statements from a specified XML file, QPTune can undo previous fixes to the server. Specify the name of the XML file and the target database, and click Next for the Options page.

  On the Options page, specify a threshold for count of missing statistics. The default threshold count is 5. See "Collecting statistics" on page 21 for more information on threshold for count of missing statistics.

  When you click Finish, QPTune goes through the steps to undo the previous fixes for missing statistics. QPTune then displays a summary page of commands that the wizard issues to it.

To use the Fix Missing Statistics or Undo Missing Statistics Fix wizards, you must have sa_role and sso_role. For more information on QPTune's cycle for fixing missing statistics, see "Using QPTune to fix missing statistics" on page 19.

Note You must run the application at least once before QPTune can collect or fix missing statistics.

Tuning Task

QPTune includes a panel called Tuning Tasks that displays existing tuning tasks on every qualifying server. A Wizard guides you through QPTune's tuning cycle. The definition of a tuning task is stored on the client machine where ASEP runs, and may only be accessed on that machine.

QPTune includes several stages in the tuning cycle for applications or queries. For more information on QPTune's cycle for application or query tuning, see "Using QPTune to tune queries or applications" on page 24.

To create a new tuning task for a server:

- Connect to the server on which you wish to tune your query.

- Click on your server name. You see the Tuning tasks tab.

  Note If you do not see the Tuning Task tab, please check that your environment variables are set correctly and that your installation contains all the required files and directories.

- Click on the Tuning Tasks tab, and then right-click in the window to bring up the “New”-> “TuningTask” menu item.
- Select the “New”-> “TuningTask” menu item. The QPTune Wizard opens.

Alternately, you can bring up the Wizard using the “Tuning Task” creation button that is provided on the toolbar.

Note You must have sa_role and sso_role to use the menu item and the creation toolbar button.

The QPTune wizard includes these screens corresponding to the different stages in tuning the Adaptive Server:

- Setup:
  - Name and Configuration

    Specify the task name and the configuration file associated with the task. When these are both specified, the Next and Finish buttons ar enabled. If the configuration file already exists, the wizard indicates this by displaying a note under the file name.

    You may select the Verbose Mode option to generate more detailed output.
  - Server Configuration

    You can view or edit server configuration commands issued during the different stages of running QPTune. Changes to the commands are written to the configuration file right before execution on the Comparison page.
  - Mode Selection

    You may select different modes to run QPTune. All three pre-programmed modes are selected by default:
    - Decision Support System (DSS)
    - Online Transaction Processing (OLTP)

- Mixed Workload (MIX)

  To define a customized mode, click the Add button. To change the order of the modes, use the Up and Down buttons. Two or more modes must be selected for the tuning tasks in order to ensure at least two collected results for later comparison.

  A customized mode is a collection of tuning parameters grouped under an optimization goal for a set of queries. The OK button is enabled only if a name, at least one rule, and the result file are specified.

  To add or edit a rule, use the pop-up text input box. To delete a rule, use the Remove button.

- Collect:
  - Application

    You can specify an executable or a script file to include before the collection phase begins.

  - Collection

    You can specify collection settings on this page. By default QPTune only collects optimization goal settings with no collection delay, and evaluates the average elapsed time for collection.

- Compare:
  - Comparison

    Specify comparison threshold setting (percentage and absolute values) and the output filename on this page.

    Click Finish to save the task definition and the configuration file.

    If you click Execute, QPTune executes all the specified modes, collects the metrics, and compares and saves the results into the output file. You may use the Preview button to list the commands that are about to be issued.

  - Results

    This page displays the output of the tuning process. Comparison results depict the performance improvement if the best plans are chosen for each of the queries. The output XML file contains the best plans or optgoal settings for each of the queries.

You may also apply the fixes to the server: Click Fix to apply the best plans or optgoal setting to the queries. This generates entries in the sysqueryplans table for the queries that are being fixed.

Select Apply Default Optimization Goal if you want to apply the default optimization goal to the server during the Fix operation. The default optimization goal is the optgoal setting that most queries selected as their best optgoal during the Compare operation.

If Apply Default Optimization Goal is selected, subsequent Fix operations apply the best result to the rest of the queries that have not selected this default optgoal setting as their best optgoal.

If Apply Default Optimization Goal is not selected, the Fix operation is applied to all the queries in the result file.

QPTune reference information

Description

QPTune is an Adaptive Server utility written in Java/XML. It enables users to identify the best query plan, optimization goals, or other configuration settings, and apply them at the query level. This results in optimal performance of later query executions.

Syntax

QPTune [-U <username>] [-P <password>] [-S <hostname:port/database>]
[-A <action [start|collect(_full)|compare|fix|(start|collect|fix|undo_fix)_stats]>]
[-M <mode>] [-T <appTime>] [-i <inputFile>] [-o <outputFile>]
[-f <fileList(,)>] [-c <configFile>] [-l <limit>] [-e <evalField>]
[-d <diff%(,diff_abs)>] [-m <missingCount>] [-n <login>] [-J <charset>]
[-N (noexec)] [-g (applyOptgoal)][-v (verbose)] [-s (sort)] [-h (help)]

Example:

```
QPTune -U sa -P -S my_host:5000/my_database -A collect
-M allrows_mix -T 0
-o metrics.xml -c config.xml -e elap_avg -d 5,5 -l 5 -
i metrics.xml
-fa1.xml,a2.xml,a3.xml -v –s
```

Parameters

-U *username*

specifies the database user name.

-P *password*

specifies the database password.

-S *server*

specifies the database server. The database server is denoted by *host*:*port*/*database*.

Note You must specify the -S option while using any QPTune actions.

-A *action*

specifies the action to be taken. One of: start | collect | collect_full | compare | fix | start_stats | collect_stats | fix_stats | undo_fix_stats.

-J *charset*

specifies the character set used to connect to Adaptive Server. If this option is not specified, the Adaptive Server uses the server's default character set.

Note If the installed JRE does not support the server's default charset encoding, you see an error message during the login process. Use the -J option to specify a more generic character set, such as -J utf8.

-M *mode*

specifies the optimization goal or custom mode for an application. One of: allow_oltp, allow_dss, allow_mix. You may also define custom modes.

-T *appTime*

specifies the application running time, in minutes.

-o *outputFile*

specifies the output file.

-i *inputFile*

specifies the input file for the fix action. You can also use -i to apply special rules to the specified queries for the start and collect for custom modes.

-f *fileList*

compares a list of files to get the best plans; use commas to separate filenames.

-c *configFile*

specifies the configuration file.

-l *limit*

specifies a limit on the number of queries that should be analyzed and applied with special rules.

-e *evalField*

evaluation field used for performance comparison.

-d *difference*

specifies the percentage and absolute value difference for performance improvement to be considered outstanding.

-N

used along with fix_stats and undo_fix_stats, -N generates a SQL script with update statistics or delete statistics statements. The update or delete statements are not executed through QPTune. The statements are written into a SQL script that is specified by the -o option.

-n *login*

specifies the user's login whose query executions are collected and analyzed.

-m *missingCount*

specifies the threshold value for missing statistics. The default value is 5.

-v

specifies verbose mode.

-g

when used along with the fix action, applies the default goal. The default goal is the best optgoal setting that most queries used as the best plan using QPTune's fix action. This option only generates plans for queries that do not currently use the server's default optimization goal.

If specific values are not indicated for the parameters, the following defaults are used:

- -A : collect
- -M : allrows_dss
- -T : 0
- -o : metrics.xml
- -c : config.xml

- -e : elap_avg
- -d : 5,5. If only percentage is specified, absolute value defaults to 0.
- -l *limit*
- -m 5

Permissions

Only users with sa_role and sso_role can run actions other than compare on QPTune.

CHAPTER 3

Running the Query Processor in Compatibility Mode

Adaptive Server version 15.0 includes substantive changes to the query processor. For most customers, the new query processor provides a faster and more efficient environment. However, you may have tuned your server and applications based on the more restricted query processor from Adaptive Server version 12.5.4 and earlier and find the benefits of the version 15.0 query processor unsuitable in some situations. In that case, use the compatibility mode to upgrade to Adaptive Server 15.0 from version 12.5.x but retain performance characteristics similar to version 12.5.x. When you enable compatibility mode, Adaptive Server 15.0 uses a query engine similar to the one used in version 12.5.4, and provides an alternative optimization and execution strategy.

Enabling compatibility mode

On Adaptive Server 15.0.3 ESD #1 and later, you can enable compatibility mode at the session or server-wide level:

- Session level – use `set compatibility_mode on | off` to enable or disable compatibility mode for the current session.
- Server-wide – use `sp_configure 'enable compatibility mode', 1 | 0` to enable or disable compatibility mode for the server.

Setting compatibility mode at the session level takes precedence over the server level.

enable compatibility mode is a dynamic configuration parameter; you need not restart Adaptive Server for it to take effect. However, you must remove any compiled plans for stored procedures, or ad hoc queries, from the statement cache.

Note sp_configure generates warnings to indicate that enabling compatibility mode does not affect cached query plans that are already in the procedure or statement cache.

sp_configure also generates warnings if it detects configuration options that conflict with compatibility mode, such as:

- One of abstract plan dump, abstract plan load or abstract plan replace is set.
- statement cache and literal autoparam are set.
- The histogram tuning factor is set to a value other than 1.

Feature support in compatibility mode

Once you enable compatibility mode, Adaptive Server uses the query processor for all insert, delete, update, and select queries.

The query processor uses either full or partial compatibility mode:

- Full compatibility mode – Adaptive Server 15.0 uses an optimization and execution strategy similar to the one used in version 12.5.x.
- Restricted compatibility mode – Adaptive Server uses only an optimization strategy similar to the one used in version 12.5.x.

Generally, Adaptive Server uses full compatibility mode wherever possible. If it cannot use full compatibility mode, it switches to restricted compatibility mode.

Table 3-1 lists restricted compatibility mode support for features in Adaptive Server versions earlier than 15.0:

Table 3-1: Version 12.5 feature support in compatibility mode

12.5.4 features with limited support in compatibility mode	Supported in full compatibility mode?	Supported in restricted compatibility mode?
Queries with text and image columns	No	Yes
Referential integrity	No	Yes
Inserts that require referential integrity	No	No
Proxy tables	No	Yes
Round-robin partitions	No	No
Encryption and cipher text	No	Yes
Queries that include the rand2 function	No	Yes
Abstract plans, either explicit (with plan clause) or implicit (plan dump or plan load);	No	Yes
Extended datatypes, such as Java ADT and Java UDF	No	Yes
XML functions	No	Yes
Browse mode	No	Yes
Parallel hints	No	No
Parallel sort	No	No

Compatibility mode does not support these Adaptive Server 15.0 features:

- Partitioned tables
- group bys with more than 31 columns
- Scrollable and insensitive cursors
- Commands on computed columns
- Queries that fire “instead-of-triggers”
- Queries executed inside “instead-of-triggers”
- Queries that issue parameterized literals in the statement cache, unless the query includes an insert...values command
- Query processing diagnostics used by showplan_in_xml
- Queries that include hash or hashbyte functions
- User-defined functions (SQL UDFs)

- Explicit timestamp inserts (available for Adaptive Server version 15.0.2 and later and in Replication Server)
- SQL-based replication (available for Adaptive Server version 15.0.3 and later)
- group by result rows that are wider than the maximum size that fits on a data page
- xmltable function

Note Query plans in compatibility mode are not executed as parallel plans.

Additional trace flag for diagnostics

Trace flag 477 alters compatibility mode. For every query evaluated, Adaptive Server prints this message to the error log, which indicates if Adaptive Server used full compatibility mode to process the query:

```
Compatibility = true | false
```

The message includes the reason if compatibility mode is not chosen, and the query text, if available.

New stored procedure *sp_compatmode*

Use sp_compatmode on Adaptive Server 15.0.3 ESD #1 and later, to verify if full compatibility mode can be used effectively. sp_compatmode generates warnings if it detects configuration options that conflict with compatibility mode, such as:

- One of abstract plan dump, abstract plan load or abstract plan replace is set
- statement cache and literal autoparam are set
- The histogram tuning factor is set to a value other than 1

Example 1: Execute sp_compatmode with these server options:

- compatibility mode is set

- dump/load/replace is “on”
- statement cache is “on”
- literal autoparam is “on”
- histogram tuning factor is set to 20

```
1> sp_compatmode
Compatibility mode is enabled.
WARNING: Compatibility mode will not be used when
'abstract plan dump/load/replace' is on.
WARNING: Compatibility mode may not be used when
statement cache and literal autoparam are enabled.
WARNING: The configuration option 'histogram tuning
factor' is configured with value '20', which is not the
default value in ASE 12.5. This may lead to different
accuracy of statistics and different query plans.
(return status = 0)
```

Example 2: Execute sp_compatmode when compatibility mode is not set:

```
1> sp_compatmode
Compatibility mode is not enabled.
(return status = 0)
```

Note Changing the configuration of the histogram tuning factor from the default in Adaptive Server 15.0 (20) to the default in Adaptive Server 12.5 (1), creates plans that are more consistent with Adaptive Server 12.5.

Changes to @@qpmode global variable

In compatibility mode, *@@qpmode* displays the query processing mode in which the previously executed query was processed. There are four query processing modes:

- 0 – a query that cannot be optimized; for example, create table, set, and so on.
- 1 – a query executed in full compatibility mode.
- 2 – a query executed in restricted compatibility mode.
- 3 – a query executed with the 15.0 query processor.

Diagnostic tool

set showplan output displays the query plan in a format similar to Adaptive Server 12.5.4, provided that the query is processed using full compatibility mode.

Index

Symbols

Numerics

A

B

C

D

E

F

G

L

M

O

P

Q

R

S

T

U