Infor LN Performance, Tracing, and Tuning Guide for Oracle

Publication Information

Release: Infor LN 10.x

Publication date: November 28, 2014

Document code: B0078C

Contents

About this guide

This document provides guidelines to improve the performance of an Infor LN environment on an Oracle database by tracing and tuning the environment. The following chapters describe the processes to improve the database and the Infor LN application.

All information is based on the use of the Infor LN software.

Note: This document is a comprehensive compilation; however there may be instances wherein relevant information or procedures may have been omitted. Therefore, we strongly recommend verifying the proposed changes in a test environment before moving to production. The information provided may not hold true for future versions of the Oracle database.

Intended audience

This document is intended for intermediate to expert Infor LN and database Administrators and Technical Consultants to get an optimal performance from an Infor LN system.

Related documents

Sections in this document are described in additional detail in other documents. These documents help to extend the knowledge in particular areas:

- *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*
- *Infor LN - Sizing guide (B0045 US)*
- *Infor LN - Data compression (B0050 US)*
- *Infor LN Installation Guide (U9498 US)*
- *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US)*

You can find the documents in the product documentation section of the Infor Xtreme Support portal, as described in "Contacting Infor" on page 8.

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at www.infor.com/inforxtreme.

If we update this document after the product release, we will post the new version on this Web site. We recommend that you check this Web site periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Chapter 1 Tuning Oracle for Infor LN

1

The performance of Infor LN also depends on the database performance. This chapter details the important areas of Oracle that require tuning.

Introduction

There is no standard method to improve the performance of your system. This chapter describes the required actions, based on the performance issues you have identified. Every new version of a database includes new settings, tools, and so on. New parameters are also introduced in each version, and some defaults are modified. Most environments run efficiently with an out-of-the box Oracle database but some environments require specific tuning for optimal performance. These topics are discussed:

- I/O setup
- SGA Setup
- Locking
- Creating statistics
- SQL Plan management
- Keep data in buffer cache
- Oracle parameters
- Data compression

Note: In addition to tuning the Oracle database, you must ensure that the current Oracle updates and patches are installed.

For more information about Oracle database tuning, See https://docs.oracle.com/en/database/database.html. The basic tuning of Oracle depends on the version used. Important quick reference guides can be found at the following Oracle knowledge base articles:

- 390374.1: Oracle Performance Diagnostic Guide
- 248971.1: Query Tuning Best Practices
- 68735.1: Diagnostics for Query Tuning Problems
- 67983.1: Oracle Net Performance Tuning

The recommended Oracle performance tools are the Oracle Enterprise Manager, statspack, AWR, and ADDM. The Oracle ADDM process creates a default snapshot every hour for easy comparison

(for a period of time). However, snapshots older than a week are deleted. Therefore, it is recommended to change the setting using the following command:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.modify_snapshot_settings(
    retention => 43200,      -- Minutes (= 30 Days)
    interval  => 30          -- Minutes.
  );
END;
```

I/O Setup

The *Infor LN - Performance, Tracing and Tuning Guide (U9357)* provides the guidelines for a correct I/O setup. For Oracle, the following additional guidelines are important:

- Allocate adequate size for the Undo tablespace to prevent “latch: undo global data” concurrency. The minimum recommended starting value is 10 GB. In case, the “latch: undo global data” is displayed in the AWR report, the size of the Undo tablespace must be increased even if sufficient free space in the Undo tablespace is available.
- For Infor LN, the minimum recommended size is 1 GB for 6 redo log files, to prevent log writer wait events. For large environments redo logs of 10 GB are not uncommon.
- During the Infor testing process, there were no major performance differences when using a filesystem for storing the database files or using the Oracle Automatic Storage Management (ASM). See Oracle support knowledgebase article 1187723.1. It is recommended to use ASM on HP-UX and Linux to use asynchronous disk I/O. See the description of the disk_asynch_io parameter in the “Oracle parameters” section.

SGA Setup

The following rules provide a general guideline for the Oracle SGA size:

- If the system is only used for the Oracle database, 40% of the physical memory can be allocated to Oracle shared memory.
- If the system is used for Infor LN and the Oracle database, 25% of the physical memory can be allocated to Oracle shared memory.

See *Infor LN - Sizing Guide (B0045 US),* for additional guidelines about the expected memory usage of the Infor LN system.

The Oracle (shared) memory must be divided into different areas. In general, if more control is provided to Oracle on the memory parameters, the more resources are utilized.

You can use the memory_target parameter for automatic memory tuning, but for performance reasons, Oracle recommends the use of sga_target and pga_aggregate_target.

Locking

In a multi-user environment, the performance of the end user can be affected by locked records or tables. Usually, long duration of locks occur due to an application issue. When locking occurs on a table that starts with ttcmcs050, ensure that first free number caching is enabled in Infor LN. See *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*.

To troubleshoot locking issues, information can be gathered using the Oracle alert log files and trace files. Oracle detects deadlocks and the Oracle Enterprise manager can provide additional information such as which user is blocked by another user. For more information about locking concepts, navigate to https://docs.oracle.com/database/121 and search for “Data Concurrency and Consistency”.

Creating statistics

Statistics on Oracle tables and indexes can be generated with the dbms_stats packages. See Oracle support knowledge base article 1445302.1. Using the following command you can generate statistics for tables and indexes:

```
SQL> execute dbms_stats.gather_table_stats('INFORLN','TTTTXT010000', cascade => true);
```

It is also possible to create statistics for the whole schema:

```
SQL> execute dbms_stats.gather_schema_stats('INFORLN', cascade => true);
```

The estimate percentage helps to overrule the standard percentage of blocks that are scanned. The statistics are more accurate when the estimate percentage is higher. However, the duration to generate these statistics is longer. Adding the “cascade => true” extends the statistics with index statistics. Statistics can be generated, automatically by a job or manually, during the creation of the tables. The automatic update of statistics differs from the manual process, which can affect the performance. For example, if statistics are updated by a script during the weekend, the level of performance decreases during the week. You can then consider disabling the Oracle job used to create these statistics, and only run the manual script at a regular interval. The job can be found using the Enterprise Manager. Go to the Server Tab > Scheduler Jobs > All > MGMT_STATS_CONFIG_JOB.

When manually creating statistics, Oracle recommends using the automatic sample_size option for statistics, as several algorithms are turned off when the sample size is set.

To measure the impact of updating the optimizer statistics before implementation or when implementing newly created statistics for all the dependent objects at once, use the “pending statistics” option. See Oracle support knowledge base article 1456776.1.

In addition to creating statistics for tables and indexes, it is also recommended to create system statistics. System statistics enable the query optimizer to accurately estimate I/O and CPU costs, enabling the query optimizer to choose a better execution plan. To create the system statistics, use the dbms_stats.gather_system_stats package. See Oracle support knowledge article 149560.1.

SQL Plan Management

You can revise the standard Oracle execution plan by using the Oracle Enterprise manager or SQL*Plus. This option can be very useful when optimizing the performance of specific queries. See Oracle support knowledge base article 456518.1.

Retain data in the buffer cache

The Oracle buffer cache can be trashed by bad queries, import or export sessions, or batches that are not scheduled as required. To retain information in the cache, the buffer_pool keep option must be used:

```
ALTER TABLE <tablename> STORAGE (BUFFER POOL_KEEP);
ALTER INDEX <indexname> STORAGE (BUFFER_POOL KEEP);
```

These tables can be retained in memory::

- Frequently used static tables and indexes.
- Small tables that are frequently accessed.

Examples of small tables are currencies, units, and business partners.

Oracle parameters

As Infor LN is mainly an OLTP oriented application, the virtual machine stops retrieving rows after the first rows are fetched for multiple queries. However, Oracle generates an execution plan for the whole result set. This execution plan is not always optimal for Infor LN. For most queries, the NESTED_LOOPS algorithm is preferred over HASH joins, SORT's, and VIEW's, because this algorithm retrieves the first set of records faster. To enable the Oracle optimizer so that NESTED_LOOPS is the preferred option, use parameters such as db_file_multiblock_read_count, optimizer_index_caching, _optimizer_sortmerge_join_enabled and _hash_join_enabled.

The table shows the Oracle parameters for which benchmarks and customer experience showed an improved performance. Some of the parameters are optional, while other parameters are recommended. All the parameters are explained in this chapter. Note: The "underscore" or "hidden" parameters are data dependent and must be tested thoroughly. See Oracle support knowledge base article 315631.1, for more information on Oracle hidden parameters.

Parameter	Value	Importance	Remarks
HP-UX_SCHED_NOAGE	178	Recommended	HP-UX systems only. Requires OS privileges.

Parameter	Value	Importance	Remarks
lock_sga	TRUE	Recommended	See the platform specific documentation.
use_large_pages	ONLY	Recommended	See the platform specific documentation.
_enable_NUMA_support	TRUE	Recommended on NUMA systems	Tested on Oracle 11.2.
filesystemio_options	SETALL	Recommended	
disk_asynch_io	FALSE	Required	Platform dependent
recyclebin	OFF	Recommended	Disable when not used.
db_file_multiblock_read_count	-	Recommended	Retain default value.
optimizer_index_caching	0 or a value between 10 and 90	Recommended	Test with both values.
optimizer_index_cost_adj	10	Recommended	
_always_semi_join	NESTED_LOOPS	Optional	Use only in specific cases.
_hash_join_enabled	FALSE	Optional	Use only in specific cases.
_optimizer_sortmerge_join_enabled	FALSE	Optional	Use only in specific cases.
_optim_peek_user_binds	FALSE	Optional	Use only in specific cases.

Caution: Prior to implementing any of the mentioned parameters, it is recommended to study the parameter behavior in the Oracle documentation.

Setting these parameters in the database can affect other products or other Infor LN sessions in the same database. However, you can to set these parameters only for the Infor LN users, by defining the same in the db_resource using the ora_alter_session option:

```
ora_alter_session:set "_optim_peek_user_binds"=false optimizer_index_cost_adj=10
```

For testing new settings, the ora_alter_session is the best solution, because non-dynamic parameters can be easily validated and a restart of the database is not required. If the tests show improvements, the values can be moved to the db_resource file of all users; if the database is only used for Infor LN, the parameters can be set at the Oracle level. A specific test db_resource file can

be created and tested by setting USR_DBS_RES. See “Tuning Infor LN” in the *Infor LN Performance, Tracing and Tuning guide (U9357 US).*

Modified Oracle parameters are logged in the alert.log file, and can be viewed using the Enterprise Manager. The current used parameters can be listed in SQL*Plus using:

```
SQL> show parameter
```

The available normal and hidden parameters can also be listed using the following query:

```
col parameter for a50
col description for a70
col value for a20

set linesize 160
set pagesize 999

select i.KSPPINM parameter,
       v.KSPPSTVL value,
       i.ksppdesc description
from   x$ksppi i, x$ksppcv v
where  i.indx = v.indx
order by ksppinm;
```

Unicode and Multi language Enabling (MLE)

Customers using the Unicode mode with MLE must monitor the Oracle parameters, _hash_join_enabled and _optimizer_sortmerge_join_enabled. See InforXtreme solution 813406 - Bad performance with UNICODE and MLE with Oracle database.

From Oracle 12c onwards, the definition for some Infor LN tables, a db_block_size of at least 16kB is required, when in the Unicode mode. See “Pre-installation tasks” in the *Infor LN Installation Guide (U9498 US)*. Also, the collation settings of the Oracle 12c database (when in the Unicode mode) must be modified. See Infor Xtreme solution 22853480.

HP-UX_SCHED_NOAGE

To allow the Oracle Database to use the SCHED_NOAGE scheduling policy (using the Oracle initialization parameter HPUX_SCHED_NOAGE=178, which is the default from Oracle 11g onwards). The OSDBA group (typically, the dba group) must have the RTSCHED and RTPRIO privileges to modify the scheduling policy, and to set the priority level for the Oracle processes. To grant the dba group the required privileges:

1. Add a line "dba RTPRIO RTSCHED" to "/etc/privgroup" file.

   Note: The group dba assumes the oracle owner's ID is part of the dba group.

2. As root, run the following command:

   ```
   /etc/setprivgrp -f /etc/privgroup
   ```

3 Start Oracle database.

Lock SGA into memory

To improve performance and to prevent the Oracle SGA paging out of memory, the SGA can be locked to the memory. To lock the SGA in the memory, the Oracle user requires special OS. The LOCK_SGA=TRUE parameter is platform specific, see the platform specific documentation. It is recommended that you use the LOCK_SGA parameter, in combination with large/huge pages. Also, use the USE_LARGE_PAGES='ONLY' parameter, to ensure that all large pages can be allocated when the database is started.

AIX

To support LOCK_SGA=TRUE on AIX, run the following command as user root:

```
chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE oracle
```

HP-UX

On the HP VM, the virtual address mappings and teardowns, along with the emulation of the translation table can get expensive, so on a HP VM the Oracle SGA must be locked to the memory.

To support LOCK_SGA=TRUE on HP-UX:

1 Add line "dba MLOCK" to "/etc/privgroup" file.

Note: The group dba considers the oracle owner's ID as part of the dba group.

2 As root, run the following command:

```
/etc/setprivgrp -f /etc/privgroup
```

3 Start Oracle database.

Linux

You must enable the HugePages functionality in Linux to use LOCK_SGA for Oracle. See *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*.

Windows

To allow the use of large pages in Windows, ensure that the Windows registry keys ORA_LPENABLE and ORA_LPSIZE are defined as a string (REG_SZ) datatype. To enable large page support in Windows, use the Oracle Windows registry entry for ora_lpenable:

```
"ORA_LPENABLE" = "1"
```

Oracle Solaris

The LOCK_SGA=TRUE setting parameter is not supported on Solaris. See Oracle support id 121983.1.

_enable_NUMA_support

Based on benchmarking, NUMA (Non Uniform Memory Architecture) enabled systems can benefit using the NUMA hardware on Oracle.

A NUMA based system is segregated to nodes, and a NUMA-node is allocated for each multi-core socket. The main system memory is divided across these nodes. Local memory (memory on the node) can be easily accessed. However, to access memory on other node(s) you must use the cross-node link, which translates to additional latency and issues.

There are various recommendations for Oracle and NUMA. It is recommended that you see the hardware and operating specific guides. The basic recommendations:

- Do not start the tuning process, if the customer requirement is small. This allows you to avoid additional complexity. The Oracle is not affected.
- With NUMA support enabled, Oracle divides the SGA across the NUMA nodes. Some systems can reserve a part of the main memory, which can be accessed from all nodes with the same access time. This is also called Interleaved Memory (ILM). Oracle stores a part of the SGA to ILM for shared access. For HP-UX based systems, the default setting MostlyNUMA for BIOS memory interleaving must be retained.

Note: compared to a non-NUMA or UMA system, additional total memory is required for Oracle to meet the per NUMA node memory requirements.

For more information about NUMA technology, see:

- http://en.wikipedia.org/wiki/Non-uniform_memory_access
- IBM Power systems performance guide http://www.redbooks.ibm.com/redbooks/pdfs/sg248080.pdf
- HP whitepaper “Red Hat Linux NUMA Support for HP ProLiant Servers”
- HP whitepaper “Locality-Optimized Resource Alignment“
- Oracle support note 864633.1 “Enable Oracle NUMA support with Oracle Server Version 11gR2”

Caution: It is recommended that prior to enabling the NUMA support, sufficient testing must be carried out before the production.

filesystemio_options

You can use the filesystemio_options initialization parameter to enable or disable asynchronous I/O or direct I/O on the File System files. This parameter is platform-specific and the default value varies by database version and operating system. This setting has no effect when Oracle ASM is used.

During benchmarking, the best results were obtained using the value SETALL, which uses both direct I/O and asynchronous I/O when possible. To set filesystemio_options, run the following command and restart the database:

```
SQL> alter system set filesystemio_options=SETALL scope=spfile;
```

For more information navigate to https://docs.oracle.com/database/121 and search for "filesystemio_options".

disk_asynch_io

To benefit from asynchronous disk I/O it is recommended to implement Oracle Automatic Storage Management (ASM). For more information navigate to https://docs.oracle.com/database/121 and search for "ASM".

When the Oracle data files are placed on a file system, you must follow the guidelines for disk_asynch_io provided by Oracle for your platform. For more information navigate to https://docs.oracle.com/database/121 and search for "disk_asynch_io".

When running Oracle on Linux and HP-UX, ensure that you set the disk_asynch_io initialization parameter to FALSE if the file system is used for storing the database files. By default the value of disk_asynch_io is TRUE.

Disable Oracle recycle bin

The use of the Oracle recycle bin can cause performance or administration issues. If you are using the recycle bin and there are performance issues, the recycle bin can be disabled:

```
alter system set recyclebin=OFF scope=both;
```

db_file_multiblock_read_count

The db_file_multiblock_read_count parameter can be used to minimize I/O during table scans. The value of db_file_multiblock_read_count can has a significant impact on the overall database performance and it is not easy for the administrator to determine the most appropriate value. Do not change the default value of this parameter.

optimizer_index_caching

This parameter is used to control the cost analysis of an index probe with a nested loop. The range of the values 0-100 indicates the percentage of index blocks in the buffer cache, which modifies the optimizer assumptions with regards to index caching for nested loops and IN-list iterators. If the value is 100 this means that 100% of the index blocks are found in the buffer cache, so the optimizer

adjusts the cost of an index probe or nested loop accordingly. It is recommended that se you use caution when setting this parameter as the execution plans can change for index caching. If the execution plan displays hash joins while nested loops can improve performance, it is recommended to conduct a test with a value higher than 0 (for example, 90).

optimizer_index_cost_adj

The optimizer_index_cost_adj parameter can be used to adjust the cost of index probes. The range of the values is 1 to 10000. The default value is 100, which means that indexes are evaluated as an access path, based on the normal costing model. A value of 10 indicates that the cost of an index access path is one-tenth of the normal cost of an index access path. Most Infor LN environments benefit if this parameter is set to a lower value. The value 10 is commonly used for Infor LN environments.

_always_semi_join

Change the value of "_always_semi_join" to NESTED_LOOPS, to optimize bad performing queries containing the EXISTS/IN condition where the execution plan contains a HASH optimization in the execution path. The default value for this parameter is CHOOSE.

_hash_join_enabled

When set to TRUE, the hash joins are enabled, for which a hash table is created on the join key of the smallest table; the other tables are then included to find a match. For Infor LN, it is recommended to set this value to FALSE, because Infor LN breaks down a lot of queries after a few fetches. For Cognos, the TRUE value can result in a better performance large ranges of data are processed.

_optimizer_sortmerge_join_enabled

Sort merge joins perform better than nested loop joins for large data sets. However, in most cases, Infor LN only considers the first few rows of the result set. Therefore, it is recommended to set this parameter to FALSE, when sort merge joins are identified in the query execution plan. When set to FALSE, the Oracle optimizer selects the nested loops instead of the sort merge joins.

_optim_peek_user_binds

When set to TRUE, the optimizer uses the value of the bind variables, when the query execution plan is compiled and no plan exists in the memory. When the query execution occurs with a full-range setting, the performance of the query is acceptable. However, the next time a “closed range” is selected, the previously generated execution plan may not be optimal. This can result in varying response times for the same selection. To avoid bind variable peeking, _optim_peek_user_binds can be set to FALSE. Note: The tkprof output can be misleading because EXPLAIN PLAN FOR command is not reading the actual data in the cache for the query. See Oracle support knowledgebase article 387394.1.

Data Compression

Data can grow faster than expected; especially when history is stored for extended periods or if the financial logging option is used. From porting set 9.0 onwards, Infor LN supports advanced table and index compression. The benefits of data compression:

- Significant savings in disk storage space.
- Fewer page reads, because more rows can fit on a page.
- Reduced disk I/O activity, because more compressed rows than uncompressed rows are included in a page.
- Ability to compress older data that is not accessed often, such as archived tables. The frequently accessed data can be stored in the uncompressed form.
- Possibility to free space no longer required for a table.
- Faster backup and restore capabilities.

See Infor LN Data compression (B0050 US) for more information.

Chapter 2 Tuning Infor LN for Oracle

2

This chapter describes the important Infor LN performance settings and parameters, see *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US).*

Important Oracle db_resource parameters

It is recommended to set these parameters in the <BSE>/lib/db_resource file:

```
ora_init:0101000
oracle_home:<Oracle home dir>
oracle_sid:INFORLN
ora_default_tablespace:dataspace
ora_temporary_tablespace:temp
#following parameter should only be set on a new database or after
#converting the data to varchar
ora_use_varchar:1
```

It is recommended that you use following parameter when Infor LN is used without Multi Language Enabling (MLE):

```
ansi_outer_join:0
```

Use these parameters when Infor LN uses the MLE option:

```
ora_alter_session:set "_hash_join_enabled"=false
"_optimizer_sortmerge_join_enabled"=false
```

In addition to these parameters, you can also use other parameters such as nls_comp, nls_sort and other unicode specific NLS. See “Globalization support” in the *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US),*

ora_init

The resource ora_init parameter is used to define several driver behaviors. Multiple behaviors can be selected by adding the octal values. The recommended optimal setting for Infor:

```
ora_init:0101000
```

See *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US)* for information about the values.

ora_max_array_fetch

The ora_max_array_fetch variable defines the maximum number of rows fetched immediately from the database to the driver. For optimal OLTP performance, it is recommended to retain the default value (5) for the ora_max_array_fetch parameter to avoid unnecessary process switches and network traffic.

A higher value of ora_max_array_fetch can be useful for some batch sessions or for table dumps. The value of this variable must be the same as the value of rds_full, which is only applicable for the non-combo mode.

ora_max_array_insert

The ora_max_array_insert variable defines the maximum number of rows that can be immediately inserted in the database from the driver. For optimal OLTP performance, it is recommended to retain the default value 1. A higher value of ora_max_array_insert can be useful for batch sessions or when uploading data. Uploading data with a value of 100 or higher can be up to 10% faster when compared to a value 1.

retained_cursors

The retained_cursors resource is used to set the number of inactive cursors that must be retained in the list, for reuse. After all rows are fetched, the driver transfers the inactive cursors in the cancel state to a cancel list, so that the inactive cursors can be assigned to a different query. However, a number of inactive cursors in this list are not available and are defined using the retained_cursors resource, and the default value is 50. If there are more than 50 cursors in the cancel list, and a request for a new cursor is issued, the cursor that is inactive for the longest duration is used. The link to the original query is removed and the cursor is assigned to a new query, which is used to implement the parsing and binding for the cursor. When the original query is implementing a re-execute process, the driver detects the cursor associated to another query. The driver retrieves the new cursor, and re-parses and binds the query. Increasing the value of retained_cursors leads to less re-parsing and rebinding of queries, which reduces the usage of CPU resources. However, the number of open cursors increases, which consumes more memory.

Usually, increasing retained cursors can help optimize performance, but only a small effect is observed for batch sessions. Therefore, it is recommended to retain the default value for OLTP usage. An example for batches:

```
retained_cursors:300
```

ansi_outer_join

When set to the default value 1, the Infor LN Oracle driver generates ANSI syntax queries, instead of the Oracle syntax. This ANSI syntax is required when using Infor LN with Multi Language Enabling (MLE) and for reporting tools. When you set the ansi_outer_join parameter to 0, the proprietary syntax is generated. You can view the difference in the output:

```
SELECT tdsls401.orno
FROM   tdsls401, tcibd001
WHERE  tdsls401.item REFERS TO tcibd001
AS SET WITH 1 ROWS
```

This results in the following (Non-ANSI) Oracle query:

```
SELECT /*+FIRST_ROWS(1)*/ a0.t$orno
FROM   inforln.ttdsls401300 a0,inforln.ttcibd001300 a1
WHERE  a1.t$item (+) = a0.t$item
```

When you use the query in ANSI mode, the following Oracle query is generated:

```
SELECT /*+FIRST_ROWS(1)*/ a0.t$orno
FROM  ( inforln.ttdsls401300 a0
LEFT JOIN inforln.ttcibd001300 a1 ON a1.t$item = a0.t$item)
```

This parameter can impact performance significant. In general, Infor LN customers without MLE benefit if the proprietary syntax (ansi_outer_join value set to 0). However, MLE customers benefit by using the default ANSI syntax.

ora_timeout

The ora_timeout resource is used to set the timeout value (in seconds) for a lock. If the lock does not succeed or fails within the specified time, the driver aborts the query and the transaction reverts to the db.retry point. The timeout can be set for 5 types of locks:

- Select for update
- Insert
- Update
- Delete
- Lock table

In general, a timeout of 30 seconds is sufficient. It is recommended that you use this resource carefully. When batches are executed during normal working hours and in some high concurrent OLTP workloads, it can be beneficial to increase the timeout to 60 seconds. In that case, the transaction waits longer to acquire e g the lock rather than re-executing the query from the retry point. Also, when the Infor LN sessions stop due to an error 107 or “Max retries (10) exceeded” condition, you can increase the ora_timeout value. However, in these situations, increasing the ora_timeout values is not the best solution. It is recommended that you identify and resolve the cause of the locking issue. See *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US)* for specific instructions to set this parameter.

ora_use_varchar

It is recommended to use the varchar data type to reduce data size and data growth. By default, string columns in the tables are created with fixed data. When using ora_use_varchar, the VARCHAR2 or NVARCHAR2 datatype is used. Using varchar reduces the amount of data storage and data growth. For an existing database, you must rebuild your database to use the varchar datatype. After converting the database or before creating a new database the resource must be set to 1:

```
ora_use_varchar:1
```

To convert the fixed length strings to varchar see section “Conversion from CHAR to VARCHAR2 strings” in *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US).*

Note: Besides indexes, also recreate views and triggers.

Oracle environment variables in the db_resource

For a better overview of the used settings, you can specify Oracle environment variables to the db_resource file, instead of specifying them in the database definition files. The following Oracle variables can be specified in the db_resource file:

- oracle_home
- oracle_sid
- two_task
- nls_lang
- nls_sort

Query hint generation

The Infor LN database driver for Oracle generates query hints to optimize the performance of the queries. When the resource ora_hint_no_hints is set, the driver does not generate hints, except for queries with hints explicitly added to the SQL query. A query hint is generated based on the following:

- Infor LN application query hints
- An ORDER BY matches (a part of) an index
- The query is only for 1 table and the WHERE clause matches (a part of) an index

The generation process is stopped when a database hint is generated. Therefore, if a hint is generated based on the ORDER BY clause, the hint generator does create a query for the WHERE clause.

See *Infor Enterprise Server - Technical Reference Guide for Oracle Database Driver (U7076 US)* for information about index hint weight calculation.

first_rows_hint

The Infor LN Oracle driver generates the FIRST_ROWS hint, for multiple queries. It is recommended to use the FIRST_ROWS(N) hint, where N is a number that specifies the amount of records expected to be returned. The first_rows_hint is introduced to switch between 3 flavors:

```
first_rows_hint:1    old style FIRST_ROWS hint will be generated
first_rows_hint:2    new style FIRST_ROWS(N) hint will be generated

first_rows_hint:0    FIRST ROWS hints will be suppressed. Note: the benchmark failed
                     with timeouts on this setting.
```

The default is set to 2.

Note: This parameter can have a significant impact on performance. However, the use of this parameter varies according to a customer's requirements.

Index Organized Tables

When Infor LN Multi Language Enabling (MLE) is used, it is recommended to recreate the MLE shadow tables as Index Organized Tables (IOT) to improve the performance. The tables registered in the Registered Tables with Multilanguage Fields (ttaad4137m000) session can be recreated using the Reorganize Tables" (ttaad4225m000) session. The Data and Indices check box must be selected. After the recreation process:

- The shadow tables containing the data language for the MLE fields are ordered by Index 1.
- The indexes for these shadow tables are removed.

This feature is available from portingset 8.8a.01 onwards. It is recommended to update to the latest portingset before recreating the tables.

Chapter 3 Tracing Oracle

3

In addition to the Infor LN tracing functionality, Oracle provides tools to trace an application. It is recommended that you start with a global tool. Start the detailed tracing based on the issues identified using the global tools.

Monitoring the database via Oracle Enterprise Manager (OEM)

Oracle Enterprise Manager provides the following performance monitoring options:

- The Performance tab in OEM displays a comprehensive overview of the system and database performance.
- The Top Activity displays the most expensive queries and the total load generated by the queries.
- SQL Tuning Advisor helps to identify a better execution plan.
- Advance Workload Repository (AWR) and the ADDM reports can be generated using the OEM, to identify the performance characteristics of a specific time window. To use AWR you must have a license for the diagnostic pack.
- SQL Monitor displays the most expensive queries, the actual and estimated plan, and costs.

In addition to these tools, OEM also provides other beneficial functions.

Tracing with SQL trace and TKPROF

The SQL trace facility is used to display the used queries and the output, in a structured format. In addition to the used queries, it is also possible to view the used execution plans.

The important benefits of using the SQL trace, instead of the ORAPROF, in Infor LN:

- Output includes the execution plan
- Output can be sorted using various methods

- The output files are smaller
- Number of used database blocks are specified

However, only the Oracle database query times are displayed. Therefore SQL trace will not support identifying a problem, if, for example, the network connection between the driver and database is causing the performance problem See the Oracle support knowledge base article 980711.1: *How to use SQL Trace and TKPROF for performance issues.*

Preparations for SQL trace

Before SQL traces are started, you must check this information:

Statistics level

The Oracle STATISTICS_LEVEL setting is used to specify the collection level of for the database and operating system statistics. The default setting of TYPICAL ensures that all major statistics required for database self-management functionality are collected, and this provides the best overall performance. The default value must be adequate for most environments. When the STATISTICS_LEVEL parameter is set to ALL, additional statistics are added to the set of statistics that are collected using the TYPICAL setting. The additional statistics are timed operating system statistics and plan execution statistics. It is recommended to set this parameter to 'ALL' only for the session level (with e.g. the resource `ora_alter_session:set statistics_level=all`) or temporaryly on system level.

The trace destination directory

Use the USER_DUMP_DEST parameter to specify the directory in which the traces must be stored. This parameter is normally specified in the active configuration file. It is recommended to use a directory with sufficient free space. This parameter can also be changed online using the following command:

```
SQL> ALTER SYSTEM SET USER_DUMP_DEST = <newdir>;
```

The maximum trace file size

The maximum trace file size can be specified using the MAX_DUMP_FILE_SIZE variable. If the trace file is truncated, this indicates that the value of the parameter must be increased or the USER_DUMP_DEST disk is full. This parameter can be changed online using the following command:

```
SQL> ALTER SYSTEM SET MAX_DUMP_FILE_SIZE = 200M;
```

File permissions

The Oracle user is specified as the owner when a trace file is created. On UNIX systems, you cannot read these files, if the user is not a member of the same group. Therefore, you require permissions to read these files. An alternative is to set the following Oracle parameter:

```
_trace_files_public = true
```

This ensures that the permissions are extended so that the file can be read by everybody. This is only required if users that do not have the permissions to access the trace files.

Create EXPLAIN TABLE

During the formatting of the trace output, the PLAN_TABLE table is created if the *Explain* option is used. Therefore, a valid Oracle user who can create a table is required, for example, the owner of the Infor LN tables. You can use UTLXPLAN.SQL script to create the PLAN_TABLE. The script is located in $ORACLE_HOME/rdbms/admin directory.

Statistics on tables

Oracle statistics must be generated using the dbms_stats package. See “Creating statistics” on page 11.

Permissions to start tracing

The Oracle user requires the “ALTER SESSION” privilege for tracing. This can be granted using these commands:

```
SQL> GRANT ALTER SESSION TO bsp;
```

The privilege is granted to the bsp user.

Or:

```
SQL> GRANT ALTER SESSION TO r_inforln;
```

The privilege is granted to the role for the inforln user.

Tracing with SQL trace

Usually, a SQL trace is executed when the application is started. To execute the trace:

- Set SQL_TRACE=true in the command line:
  ```
  -set SQL_TRACE=true
  ```
- Set sql_trace:true in the db_resource file. This is not recommended because all users are traced.
- Set SQL_TRACE=true in the database. This is not recommended because all users are traced.

It is recommended to trace the sessions one by one, and ensure that the bshell is closed after the process. This is because the entire output is in one trace file and can be difficult to analyze.

If the startup fails due to error 2031 (ORA-1031), the user is not allowed to run “ALTER SESSION” in Oracle. Grant the required rights to the user to resolve this issue. See “Permission to start tracing” on page 27.

It is also possible to trace an active process. Use the following commands:

Start trace:

```
SQL> EXEC DBMS_SYSTEM.set_sql_trace_in_session(sid=>123, serial#=>1234,
sql_trace=>TRUE);
```

Stop trace after a specific duration:

```
SQL> EXEC DBMS_SYSTEM.set_sql_trace_in_session(sid=>123, serial#=>1234,
sql_trace=>FALSE):
```

The SID and SERIAL# can be retrieved from V$SESSION on user level:

```
SQL> SELECT sid, serial#
  2> FROM   v$session
  3> WHERE  osuser = 'bsp';
```

The SID and SERIAL# can also be retrieved from V$SESSION on session level:

```
SQL> SELECT sid, serial#, module
  2> FROM v$session
  3> WHERE module = ‘<session_name>’;
```

If additional information of the Oracle user is available, for example: the Process ID, the query can be extended with that information.

To format the trace output

The output of the trace is written to a file specified using the USER_DUMP_DEST parameter. The file name differs for UNIX and Windows systems:

- ORA_<Process ID>.trc on UNIX systems.
- ORA<Tread ID>.trc on Windows systems.

The trace file can be formatted with the TKPROF utility:

```
tkprof <trace file> <output file> explain=<user>/<password> sort=<sort option>
```

tracefile

The tracefile, which can be specified without the ".trc" extension.

outputfile

The outputfile is written to the current directory if no directory is specified. If no extension is specified, the ".prf" extension is added.

explain

Used to determine the execution plan for each SQL statement in the trace file and writes these plans to the output file. `TKPROF` determines the execution plans, by issuing the `EXPLAIN PLAN` statement, after connecting to Oracle. The user and password must be specified in this parameter. The specified user must have the `CREATE SESSION` system privileges. It takes longer to process a large trace file, if the `EXPLAIN` option is used.

It is possible that the execution plan displayed is different to the plan used, because settings can be modified after login. For example, ALTER SESSION statements etc. To view the correct plan, it is recommended that you use the SQL Monitor option in Oracle Enterprise Manager.

sort

SQL statements traced are displayed in a descending order, based on the specified sort option in the output file. If more than one option is specified, the output is sorted in descending order based on the sum of the values specified in the sort options. The sort options commonly used:

- FCHELA Elapsed time spent on fetches.
- EXEELA Elapsed time spent on executes.
- PRSELA Elapsed time spent on parsing.

Other sort options can be identified using the tkprof command, without options.

table

When the default 'explain' table does not exist, tkprof uses another table to store the explain plan.

The output displays the expected application queries and other system queries; these queries are required to specify the application query. When you search for a bad performing application query, the system queries can be suppressed using the following action in the tkprof command:

```
sys=no
```

An example of using the tkprof:

```
$ tkprof ora_08456 tkprof_08456 sort=(prsela,exeela,fchela) sys=no
```

To interpret the formatted output

The interpretation of the trace output is the difficult part of a trace. However, you can use the tkprof utility to interpret the output easily. Before locating the problem, you must understand how tkprof formats the output.

For example:

```
SELECT /*+ FIRST_ROWS INDEX(b ttipcf310120$idx1)  */ a.t$preq,a.t$expl,
  b.t$mitm,b.t$pono,b.t$sern,b.t$cnsc,b.t$sitm,b.t$dsca,b.t$leng,b.t$widt,
  b.t$noun,b.t$qana,b.t$scpf,b.t$cwar,b.t$opno,b.t$cpha,b.t$exin,b.t$nnts,
  b.t$ltmo,b.t$indt,b.t$exdt,.b.t$pqan,b.t$pper,b.t$txta
FROM
 inforln.ttibom010120 a,inforln.ttipcf310120 b WHERE b.t$mitm = :1 AND (b.t$mitm =
 :2 AND b.t$pono = :3 AND b.t$sern > :4) ORDER BY 3,4,5


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       53      0.01       0.02          0          0          0           0
Execute    287      0.63       0.73          0          0        136           0
Fetch      287    198.00     238.58     930456    1070896       1632         816
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      627    198.64     239.33     930456    1070896       1768         816

Misses in library cache during parse: 1
Optimizer goal: FIRST_ROWS
Parsing user id: 613

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  FILTER
      0   MERGE JOIN CARTESIAN
      2    TABLE ACCESS BY INDEX ROWID TTIPCF310120
      2     INDEX RANGE SCAN (object id 178354)
      0    SORT JOIN
      0     TABLE ACCESS FULL TTIBOM010120
```

In the example, the query is displayed first and the header provides common information on how certain columns must be interpreted. Several rows are made for *Parse*, *Execute*, and *Fetch*.

The functions of the rows:

- **Parse**
  Prepares the SQL statement. Checks if the table(s) and the required column(s) exists and creates the execution plan.
- **Execute**
  Executes the statement. **Fetch**
  Retrieves rows returned by a query. Fetches are only performed for `SELECT` statements.

The columns for Parse, Execute, and Fetch:

- **Count**
  The number of times a SQL statement is parsed, executed, or fetched.
- **Cpu**
  The total CPU time in seconds for all parse, execute, or fetch calls for the statement.

- **Elapsed**
  The total elapsed time, in seconds, for all parse, execute, or fetch calls for the statement.
- **Disk**
  The total number of data blocks physically read from the data files on the disk for all parse, execute, or fetch calls.
- **Query**
  The total number of buffers retrieved in the consistent mode for all parse, execute, or fetch calls. Buffers are usually retrieved in the consistent mode for queries.
- **Current**
  The total number of buffers retrieved in the current mode. Buffers are retrieved in current mode for statements such as `INSERT`, `UPDATE`, and `DELETE.`
- **Rows**
  The total number of rows processed by the SQL statement.

If the columns *cpu* and *elapsed* are always zero, it is recommended to check if *timed_statistics* is set to *true*. If yes, then the the minimum resolution of timing which is 1/100 of a second, can be the probable cause.

The execution plan:

```
Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: HINT: FIRST_ROWS
      1   FILTER
      0    NESTED LOOPS (OUTER)
      1     NESTED LOOPS (OUTER)
      1      NESTED LOOPS (OUTER)
      2       TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF
                  'TFMFOC201570'
      0        INDEX   GOAL: ANALYZED (RANGE SCAN) OF
                   'TFMFOC201570$IDX6' (UNIQUE)
      0       TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF
                  'TFMFMD100570'
      0        INDEX   GOAL: ANALYZED (UNIQUE SCAN) OF
                   'TFMFMD100570$IDX1' (UNIQUE)
      0      TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF
                 'TFMFOC200570'
      0       INDEX   GOAL: ANALYZED (UNIQUE SCAN) OF
                  'TFMFOC200570$IDX1' (UNIQUE)
      0     TABLE ACCESS   GOAL: ANALYZED (BY INDEX ROWID) OF
                'TTCIBD001570'
      0      INDEX   GOAL: ANALYZED (UNIQUE SCAN) OF
                 'TTCIBD001570$IDX1' (UNIQUE)
```

The Oracle optimizer uses the execution plan to locate the result set. You must read the output from the statement on the right of the screen to the left. The tables are written below the first statement, with the indent, on the right. The standard method to use the data is above the statement.

Follow these guidelines to read trace files:

- Look for queries that are used often.
- When a query is parsed several times, it is important to check if the space for shared pool is sufficient.

- Reading a large amount of (disk) blocks must be avoided as much as possible. Therefore, it is recommended to investigate if the query can be modified to read less data blocks. However, if multiple blocks must be read, check if the buffer cache hit ratio is sufficient.
- Sometimes due to a trace file error, wrong table names are displayed in the execution plan. This can occur because several sessions write to the same tracefile. If this occurs on a query that requires additional analysis, that query can be extracted from the large trace using the tkprof and trcsess command:

Add the session's id to the trace file:

```
$ tkprof erp1_ora_20077 no_aggregate aggregate=no
```

Identify the problem query in the new trace file and write down the session ID. Create a tracefile specifically for this session ID:

```
$ trcsess session=1381.609 output=partly_trace.trc no_aggregate.trc
```

Format the trace file similar to the original tracefile:

```
$ tkprof partly_trace partly_trace sort=exeela,prsela,fchela
```

For more information about SQL trace and tkprof, navigate to https://docs.oracle.com/database/121 and search for "SQL trace".

Tracing with SQL trace event 10046

When you must trace a session using a query of another user, such as a batch job that is being executed for a longer duration, or to get additional trace information, you can use the event 10046.

The SQL trace event 10046 trace levels which are commonly used:

Level	Description
1	Include statistics for parse, execute, fetch, commit, and rollback. Same as when setting SQL_TRACE=true
4	Include values for SQL bind variables.
8	Include statistics for wait events as listed in v$event_name

These levels can be combined, but every non-zero value includes the Level 1 tracing. So, Level 12 is the same as Level 1, 4 and 8. For only query tracing, Level 1 supports sufficient information, and the information of Level 4 and 8 is not summarized correctly by tkprof. To interpret information from Level 4 and 8 traces, scripts must be written; otherwise the raw output must be interpreted.

Level 4 output can be useful to reproduce bad queries by Oracle tools such as SQL*Plus. The output of Level 8 is useful when analyzing locking issues and areas of concern with regards to general performance.

Activate or deactivate the trace

The 10046 trace can be activated from the Infor LN UI command prompt:

```
-set ORA_ALTER_SESSION="set events '10046 trace name context forever, level 8'"
```

Or the trace can be activated by finding the SID and SERIAL# from V$SESSION of the session that must be traced. For these values, the following commands must be used:

```
exec sys.dbms_system.set_bool_param_in_session(SID, SERIAL#, 'timed_statistics', true)
sys.dbms_system.set_int_param_in_session(SID, SERIAL#, 'max_dump_file_size',
2147483647)
exec sys.dbms_set_system.set_ev(SID, SERIAL#, 10046, 1, '')
...
exec sys.dbms_set_system.set_ev(SID, SERIAL#, 10046, 0, '')
```

Output of this trace is available at the same location and the format is similar to the SQL_TRACE trace.

See the Oracle support knowledge base article 376442.1: How To Collect 10046 Trace (SQL_TRACE) Diagnostics for Performance Issues.

Tracing with event 10053

Tracing with the event 10053 provides information on how the Oracle database is used to select the execution plan as identified by tkprof. All optimizer permutations are displayed, including the cost and other information. This event has only 1 level. To activate and deactivate the event, set the value to 1 or 0, respectively. See the Oracle support knowledge base article 225598.1: *How to obtain tracing of optimizer computations (EVENT 10053).*

Appendix A Format ORAPROF output

A

The output of the Oracle level 2 driver can be large, and cannot be read with all editor applications. This script formats the output to a readable format. The output columns:

- **QID**: The query ID used in this script. You can use the QID to easily locate the related query in the output, at a later stage.
- **Table**: The first table in the query.
- **Command**: Lists the following type of queries:
  - select
  - for update
  - update
  - insert
  - delete
- **Count**: The number of times the query is parsed, executed, or fetched.
- **Time**: The total amount of time the query is used for the parse, execute, or fetch processes.
- **Start Time**: The first time the query is found in the trace file.

In the original output, each query is displayed in single lines. If the output is longer than 65 characters, the queries are wrapped. This ensures that the output is readable. An example of the output:

```
                                Parse        Exec         Fetch
QID Table            Command Count  Time  Count  Time  Count  Time Start Time
  7 ttfacp610600 Select       989   1.24  1978   1.44  1978   1.94 11:35:30.750
  5 ttfacp610600 For Update     1   0.00   990   1.03   990   0.65 11:35:30.680
  4 ttfacp610600 Select         1   0.00     3   0.00     3   0.02 11:35:30.650
  1 tttadv999000 Select         0   0.00     2   0.00     2   0.01 11:35:23.450
  3 tttadv999000 Select         1   0.00     2   0.00     2   0.01 11:35:23.520
  2 tttadv112000 Select         0   0.00     1   0.00     1   0.00 11:35:23.480
  6 ttfacp610600 Update         1   0.00   990   1.00     0   0.00 11:35:30.690

Query : 1
SELECT /*+ index(a tttadv999000$idx1) */ t$pacc,t$keyr,t$desc,t$Refcntd,
t$Refcntu FROM inforln.tttadv999000 a WHERE t$pacc=:1 AND t$keyr=:2

Query : 2
SELECT /*+ FIRST_ROWS INDEX(a tttadv112000$idx1)  */
a.t$pacc,a.t$cpac, a.t$sequ,b.t$mess,b.t$expi,b.t$za_mtyp FROM inforln.tttadv112000
a,inforln.tttadv450000
b WHERE b.t$clan = :1 AND b.t$cpac = :2 AND b.t$cmes = :3 AND b.t$vers
= a.t$vers AND b.t$rele = a.t$rele AND b.t$cust = a.t$cust AND a.t$pacc
= :4 AND a.t$cpac = :5 ORDER BY 1,2,3
```

An example of the script:

```
if [ $# -ne 2 ];then
    echo "Usage $0: <inputfile> <outputfile>"
    exit
fi

Awk=awk
[ -x /usr/xpg4/bin/awk ] && Awk=/usr/xpg4/bin/awk
[ -x /usr/bin/nawk ] && Awk=/usr/bin/nawk


fold -w 800 $1 |\
$Awk 'BEGIN{
    MaxQID=0
    MAXLENTGH=65
}
{
    if (substr($1,1,5)=="------")
        getline
    if (substr($1,1,1)=="<") {
        split($0,t0,":")
        split(t0[2],t1,"[")
        split(t0[4],t2,"]")
        StTime=t1[2] ":" t0[3] ":" t2[1]
        getline
    }
    if ($2=="(parse)") {
        Time=$4
        line=NR
        getline
        getline
        getQID()
        ParseTotal[QID]+=Time
        ParseCount[QID]++
        next
    }
    if ($2=="(multi_exec)") {
        Time=$4
        getline
        getline
        getQID()
        ExecTotal[QID]+=Time
        ExecCount[QID]++
        next
    }
    if ($2=="(multi_fetch)") {
        Time=$4
        getline
        getline
        getQID()

        FetchTotal[QID]+=Time
        FetchCount[QID]++
        next
    }
}
function getQID()
{
```

```
i=0
while (substr($0,1,5)!="-----" && $0!="" ) {
   Line[++i]=$0
   getline
}
for (j=MaxQID;j>0;j--) {
   for (k=i;k>0;k--)
      if (Line[k]!=InLine[j,k])
         k=-1
   if (k==0)
      j=-j
}
MaxInLine=i
if (j==0) {
   MaxQID++
   MaxLength=MAXLENTGH
   k=1
   for (j=1;j<=i;j++) {
      TmpLine=Line[j]
      InLine[MaxQID,j]=Line[j]
      while (length(TmpLine)>MaxLength) {
            SQLLine[k]=SQLLine[k] substr(TmpLine,1,MaxLength)
         TmpLine=substr(TmpLine,MaxLength+1)
         l=index(TmpLine," ")
         m=index(TmpLine,",")
         if (m==0) m=999
         if (l==0) l=999
         if (m==999 && l==999) {
            SQLLine[k]=SQLLine[k] substr(TmpLine,1)
            TmpLine=""
         }
         else if (m>l) {
               SQLLine[k]=SQLLine[k] substr(TmpLine,1,l)
            TmpLine=substr(TmpLine,l+1)
         }
         else {
               SQLLine[k]=SQLLine[k] substr(TmpLine,1,m)
            TmpLine=substr(TmpLine,m+1)
         }
         MaxLength=MAXLENTGH
         k++
      }
      if (length(TmpLine)!=0) {
         SQLLine[k]=SQLLine[k] substr(TmpLine,1)
         MaxLength=MAXLENTGH-length(SQLLine[k])
      }
   }
   NrLines=k
   QID=MaxQID
   for (j=1;j<=NrLines;j++)
      SQL[QID,j]=SQLLine[j]
   Stat="Select"
   s=substr(SQLLine[1],1,6)
   u=substr(SQLLine[NrLines],length(SQLLine[NrLines])-5)
   if (s=="INSERT") Stat="Insert"
   if (s=="UPDATE") Stat="Update"
   if (s=="DELETE") Stat="Delete"
```

```
      if (Stat=="Select") {
         i=1
         while (index(SQLLine[i],"FROM")==0 && i<=NrLines)
            i++
         a=index(SQLLine[i],"FROM")
         if (length(SQLLine[i])==a+4) {
            b=SQLLine[++i]
         }
         else
            b=substr(SQLLine[i],a+4)
         split(b,c)
         split(c[1],d,".")
      }
      else if (Stat=="Insert") {
         split(SQLLine[1],c)
         split(c[3],d,".")
      }
      else
      {
         split(SQLLine[1],c)
         split(c[6],d,".")
      }
      if (u=="UPDATE") Stat="For Update"
      Length[QID]=NrLines
      MainTable[QID]=d[2]
      Status[QID]=Stat
      StartTime[QID]=StTime
      for (j=1;j<=NrLines;j++)
         delete SQLLine[j]
   }
   else
      QID=-j-1
}
function swap(A, i) {
   t=A[i-1]
   A[i-1]=A[i]
   A[i]=t
}
END {
   for (i=1;i<=MaxQID;i++) {
      Q[i]=i
      for(j=i;j>1 && FetchTotal[j-1]<FetchTotal[j];j--) {
      swap(MainTable, j)
      swap(Status, j)
      swap(ParseCount, j)
      swap(ParseTotal, j)
      swap(ExecCount, j)
      swap(ExecTotal, j)
      swap(FetchCount, j)
      swap(FetchTotal, j)
      swap(StartTime, j)
      swap(Q, j)
      }
   }
   print "             Parse          Exec      Fetch"
   print "QID Table     Command Count  Time  Count  Time  Count  Tim
e Start Time"
```

```
   for (i=1;i<=MaxQID;i++) {
      printf("%3d %s %-10s %5d %6.2f %5d %6.2f %5d %6.2f %s\n",
         Q[i], MainTable[i], Status[i], ParseCount[i],
         ParseTotal[i], ExecCount[i],  ExecTotal[i],
         FetchCount[i], FetchTotal[i], StartTime[i])
   }
   for (i=1;i<=MaxQID;i++) {
      print
      print "Query :",i
      for (j=1;j<=Length[i];j++)
      print SQL[i,j]
   }
}' $1
```