Quick SELECT (QSEL)

for
ORACLE™

Version 2.1.00

Reference Manual

March 2016
Log-On Ltd.

Quick SELECT (QSEL) version 2.1.00 (March 2016)
for ORACLE version 12.x, 11.x, 10.x or 9.x with Pro*C

Table of Contents

1. Introduction

1.1. General Description

Quick SELECT (QSEL) is a software tool that greatly improves the performance of what is arguably *the* most common type of SQL accesses: compiled SELECT statements. The improvement is achieved *transparently*, without the need to modify source programs. QSEL installation, operation and disabling are all very simple and easy.

The performance improvements include:

- Saving CPU time for the process that issues SELECT statements.
- Saving system overhead for interfacing with the ORACLE database server.
- Saving telecommunications overhead in case of remote databases.
- Saving CPU and I/O for the ORACLE database server, reducing its workload and thus improving the overall system-wide database performance.

1.2. Terminology

1.2.1. Update-Insensitive Tables

The term *update-sensitive table* refers to any database table from which the most up-to-date data is required upon each retrieval.

The term *update-insensitive table* refers to a table that is either not updated concurrently to retrievals from it, or that updates need not (but can) be immediately recognized.

For most of the applications that do not *update* a table, obtaining the most up-to-date data form it is not very important. For most applications it would be beneficial if data from the table could be kept in memory for performance improvement, even though this necessarily implies that the most up-to-date data is not always thus obtained. For example, a banking batch application can retrieve currency exchange rates from a database, knowing the table will not be updated concurrently.

QSEL is targeted at accessing update-*in*sensitive tables only.

1.2.2. Compiled SELECT Statements

In general, a SQL SELECT can be:

- Embedded in a source code of a *compiled* procedural programming language such as C, embedded in a source code of non-compiled programming language such as PL/SQL or in a source code of a fourth generation language, or coded in or generated by program or application generators such as PowerBuilder.
- *static* -- fully coded at compile time, or *dynamic* -- generated at run time and then EXECUTEd (sometimes PREPAREd beforehand).
- Coded as a *statement* in itself, or appear as a *clause* in another SQL statement, such as SELECT (as a subquery within it), INSERT, UPDATE or DECLARE CURSOR.

QSEL is targeted at improving the performance of static SELECT statements embedded in C programs.

Note that compiled SELECT statements are *very* common. For example, the SELECTs in a C program that retrieves values from a table having a unique key by that key.

1.2.3. Repetitive Retrievals

When accessing a table via SELECTs, the references can concentrate in a small portion of the table or, especially for small tables, even the entire table. Some of the accesses can be repetitions of previous accesses -- the same SELECT statement with the same values in the input host variables. When the number of accesses is larger than the number of accessed rows, it usually implies repetitions (the exception being when different input values still access the same row). A repetition can immediately follow the previous occurrence of the same SELECT with the same values, or be separated from it by other SELECTs and/or other input values.

Note that repetitive SELECTs are *very* common. For example, decoding bank names from a 1,000-row table for 1,000,000 input transactions.

QSEL improves performance by caching active data in memory, and avoiding database accesses in case of repetitions. Thus, the performance improvement rate is highly dependent on the rate of repetitions.

1.3. QSEL Usage Overview

1.3.1. Installation

- First, QSEL is installed.
- Then, administrator should register QSEL at Log-On Software web site, get a valid license and install it.

1.3.2. Initialization

- Then, *make*/script files for compiling/linking with QSEL are prepared.
- The administrator provides a few details to QSEL, such as which tables and JOINs to handle, how much storage to use, etc. These definitions can be overridden by a programmer for each process.

1.3.3. Usage

- Performance is improved for every SELECT statement within the QSEL scope in every program compiled/linked with QSEL. *No source program needs to be modified for obtaining the performance improvements!*
- Any program not compiled/linked with QSEL is unaffected by it. In case of separately compiled routines linked together, only those compiled with QSEL use it, and the other routines are unaffected.
- If, for whatever reason, QSEL has to be disabled, this can be done for a specific program, for a specific process, or globally for all programs.
- Only obtaining run-time statistics does require a slight modification of the source program (adding a *#include* at the logical end of the program). These statistics can be used, for example, for obtaining information about storage usage by QSEL.

1.4. QSEL Highlights

Quick SELECT provides:

- *Performance improvement*, including:
  - Saving CPU time for the *process* that issues SELECT statements, whether the SELECT is successful or results in "no data found".
  - Saving *system overhead* for interfacing with the ORACLE database server.
  - Saving *telecommunications overhead* for remote databases.
  - Saving CPU and I/O for the ORACLE database *server*, reducing its workload and thus improving the *overall system-wide* database performance.
- *Transparency* -- requiring no changes in source programs.

- *Ease* of installation, operation and disabling.
- *Control* over the use of resources and participating database tables.

With the few exceptions detailed below, the full SELECT syntax is supported, including JOINs for any number of tables, all the built-in functions and any complexity of expressions and subqueries. Host arrays are supported as well. Performance is improved not only for successful SELECTs, but also for SELECTs resulting in "no data found".

1.5. Summary of Changes

This section summarizes QSEL releases. For detailed release information, refer to "Appendix: Release Notes".

1.5.1. Version 1.1

Release 1.1 (July 1995)
- Quick SELECT now supports host arrays.
- Quick SELECT now supports performance improvements for "no data found" conditions. Note that this implies that concurrent additions to the accessed table are not usually recognized.

Release 1.12 (November 1996)
- Quick SELECT now supports Pro*COBOL.
- Quick SELECT now supports cache refresh, enabling the user to clear QSEL's cache buffers.

1.5.2. Version 1.3

Release 1.3.01 (March 2000)
Bug Fixes

Release 1.3.02 (May 2000)
- Upon initialization Quick SELECT writes a logo to the standard output.
- Whenever applicable, the application program file name and line number - where SQL statement issued - are printed as part of each diagnostics message.
- Bug Fixes

Release 1.3.03 (June 2000)
- Bug Fixes for HP port.
- Fix Quick SELECT make file to produce smaller/quicker shared library.

Release 1.3.04 (July 2000)
- Simplified the process of porting Quick SELECT to different platforms.

Release 1.3.05 (August 2000)
- Quick SELECT messages are now printed to standard output.
- Refine validation of Pro*C 8 variables.

Release 1.3.06 (September 2000)
- Extended Pro*C 8 variables validations
- Improve Quick SELECT porting to different platforms

Release 1.3.07 (February 2001)
Bug Fixed in host arrays.

Release 1.3.09 (April 2001)
- Clarified several messages by appending to them ", ID = <id>" where <id> is the statement ID assigned by Quick SELECT. This, to make problem resolution easier.
- Fixed memory overrunning bug when data is exactly 256 bytes long.

Release 1.3.10 (July 2001)
Fixed bug in pipe read.

Release 1.3.10a (August 2001)
- Disabled the Oracle server version check.

Release 1.3.11 (October 2001)
- Write the Quick SELECT logo to the "/tmp/qsel.log" file.
- Fixed bug in Pro*C 8 variables validation

1.5.3. Version 1.4

Release 1.4.02 (November 2001)
- Quick SELECT Logo is displayed if Quick SELECT actually caches or if a warning or error message needs to be displayed.
- Quick SELECT does not generate defunct children anymore.

Release 1.4.02a (November 2001)
- The logo information has been squeezed to one line.

Release 1.4.03 (November 2001)
- Quick SELECT internal buffer for output data enlarged to 64K to allow caching of large amount of data retrieved by single SQL SELECT.

1.5.4. Version 1.6

Release 1.6.02 (December 2002)
- Quick SELECT is enhanced to support Pro*C of Oracle 9.2.0
- Improve support for 64 bit platforms
- Introducing a Licensing mechanism. See more details later in this document.
- Introducing a utility to print Quick SELECT version information.
- Support AIX machines.

Release 1.6.03 (February 2004)
- Adding the SUPPRESS_LOG variable (See sections 2.2.3, 2.2.4) that allows the user to disable license messages.

Release 1.6.04 (March 2005)
- Changing the licensing mechanism to have a license created for a version work for all its revisions (E.G: licenses created for 1.6.04 will work for 1.6.05, but not for 1.7.01).
- Adding support for long source files (over 8,192 lines in .pc file).
- Adding support for long SQL statements (over 8,192 characters long).
- Adding support for accessing a non-default database.

Release 1.6.05 (October 2005)
- Fixing the mechanism to obtain IP address for license verification. The old mechanism used fork(), which caused defunc.

1.5.5. Version 1.7

Release 1.7.00 (July 2007)

- QSEL was ported to Linux. At this phase, it was built and tested on Fedora Core 4 with Oracle client 9.2.0.4 & 10.2.0.1 and Oracle Server 9.2.0.4 (on i386 platform)
- Remove QSEL limit of 65K entries in host array size.
- Remove QSEL limit of 64K bytes for cached data size.
- Remove QSEL limit of 1K bytes for total length of input host variables' values in a single statement.
- Remove QSEL limit of 65K SELECT statements per process.
- Enhancing the scope of table names candidate for caching. QSEL is now able to parse a SELECT statement text for table names not only in the first FROM clause but also in all other FROM clauses (in subqueries, unions etc.).
- QSEL statistics report was enhanced and contains new statistics about SQL activity and cache activity.
- Auto statistics report: Starting at this release, there is no need to modify application source code in order to produce QSEL report.
- A new, enhanced, verbose mode was added to allow debug information to be written to the application log file.
- Fixed bug in handling the combination of NULL columns/NULL constants with VARCHAR host variables.
- Fixed bug in handling of large SQL statements.
- QSEL Packaging: The product will be packaged for each combination of: platform , OS version, architecture. The Oracle version is no longer part of the packaging process of QSEL.
- User Guide includes an appendix that lists all runtime messages.

1.5.6. Version 1.8

- Added support for Oracle 11.1
- Fixed bug in referencing HP-UX Itanium shared-libraries with “so” suffix (during dynamic load).
- Fixed bug that caused a core dump when host name was larger than 8 characters and /etc/hosts file did not contain an entry with host name.
- Fixed bug caused to error messages to come up when license was invalid and SQL statement was long and QSEL_SUPPRESS_LOG environment variable was set to “Y”
- Support for Pro*COBOL has been **dis**continued.

1.5.7. Version 1.9

- Added support for Oracle 11.2

Version 1.9.02

- Minor syntax change in qselmc.h header to support stricter compilers
- Fixed bug where setting QSELAUST variable to Y did not produce statistics. The bug had occurred only on Solaris

1.5.8. Version 2.0

- Added support for Red Hat Enterprise Linux Server release 6.4 platform (Santiago)

Version 2.0.00

- Corrected reference to web-site in the licensing program

1.5.9. Version 2.1 (**New**)

- Added support for Oracle 12c. The initial version was produced and tested on Red Hat Enterprise Linux Server release 5.9. Regression testing performed on RHEL 5.1.
- This version was ported to SunOS 5.11 (Solaris 11) in March, 2016.
- This version was ported and tested under RHEL 7.1.

Version 2.1.00

1.6. Related Documents

- ORACLE 8.x, 9.x, 10.x, 11,x or 12.x Server SQL Language Reference Manual
- Programmer's Guide to the ORACLE Precompilers, version 8.x, 9.x, 10.x, 11.x or 12.x
- Pro*C Supplement to the ORACLE Precompilers Guide, version 8.x, 9.x, 10.x, 11.x or 12.x

2. Programmer's Guide

2.1. When to Use QSEL

2.1.1. QSEL Target Scope

2.1.1.1. Target Access Characteristics

Performance can be improved when *all* of the following are true:

- The tables are update-*in*sensitive.
- The retrievals are repetitive SELECTs issued from the same process.
- There are no run-time errors or warning conditions other than “no data found”.
- Enough storage is defined and available for keeping active data in.

2.1.1.2. Target Environment

Performance can be improved when *all* of the following are true:

- The SQL statements are coded in C programs.
- For a C program: the program is processed by the QSEL preprocessor for C and then by the standard Pro*C ORACLE precompiler.
- Control blocks and their settings as generated by the ORACLE precompiler and by the QSEL preprocessor are not modified.
- Enough storage is defined and available for keeping active data in.

2.1.1.3. Target SQL

QSEL improves the performance for repetitive retrievals via compiled SELECT statements. With the few exceptions detailed below, the full SELECT syntax is supported, including JOINs for any number of tables, all the built-in functions, and any complexity of expressions and subqueries. Host arrays are supported as well.

Any of the following inhibits the performance improvement (for the specific SELECT statement only):

1. Including a FOR UPDATE clause.
2. Using the SYSDATE built-in function.
3. Using the CURRVAL or NEXTVAL pseudo columns or referring to a table or a field named CURRVAL or NEXTVAL
4. Using an "extern char[]" host variable without a length.
5. Using a host variable with an ORACLE type (as opposed to a native C type) *other* than CHAR, CHARZ, DATE, FLOAT, INTEGER, ROWID, STRING, UNSIGNED, DECIMAL, DISPLAY, VARCHAR or VARCHAR2. Note that unsupported ORACLE types cannot be used without either EXEC SQL TYPE or EXEC SQL VAR specifying the unsupported ORACLE type. All the native C types (e.g., char, int, float) are supported. Note that the above refers only to *host variables in the C routines*, not to the type of *table columns in the database*. For example, a NUMBER table column cannot easily be handled in a C host variable declared with EXEC SQL VAR or EXEC SQL TYPE as a NUMBER. It can, however, be maintained in a host variable of, say, long int.
6. Using a host variable POINTER to a string (CHAR, CHARZ, STRING, VARCHAR or VARCHAR2), or to an unsupported type.
7. Accessing table or tables that are defined as update-sensitive.
8. Encountering any run-time error or warning condition other than "no data found".

2.1.1.4. Implication of Saving ORACLE Calls

QSEL improves performance by saving some or most of the ORACLE calls for the SELECTs supported by it. This implies the following:

1. ORACLE statistics (e.g., in the ORACA, in the trace file) do not include saved calls.
2. The *sqlgls*() function returns the text of the last SQL statement that was actually passed to ORACLE (*not* saved by QSEL). This implies that if *sqlgls*() is called after a SELECT was saved by QSEL (i.e., not passed to ORACLE), then the text of this SELECT will not be correctly returned. Note, however, that this is rarely an issue, as *sqlgls*() is used mainly by applications using *dynamic* SQL, whereas QSEL saves ORACLE calls only for *static* SQL. Moreover, *sqlgls*() is used mainly in *error* cases, usually for errors other than "no data found", as the latter are normally specifically handled by the calling program. As QSEL saves ORACLE calls only for successful and "no data found" cases, this is not normally a problem from this aspect as well.
3. In case of "no data found" access that is saved by QSEL, the fifth element of sqlca.sqlerrd (sqlca.sqlerrd[4] in C) -- the "parse error offset" -- is sometimes set by ORACLE to 1 and sometimes to 0. When QSEL saves the ORACLE call, this "parse error offset" is always set to 0.
4. The number of rows returned by a SELECT with host arrays is set in the third element of sqlca.sqlerrd (sqlca.sqlerrd[2] in C). When "sqlca" is not included when using host arrays (not generally a good programming practice), the number of rows actually returned in case of "no data found" access is unknown. In this case ORACLE leaves the host array elements past the last returned row as they were before the SELECT. When QSEL saves the ORACLE call, the values of the host array elements past the last returned row are undefined.

2.1.2. When Not to Use QSEL

In general, using QSEL for tables/accesses *not* within its scope may result in some slight degradation in performance, negligible in most cases. However, the use of QSEL must be strictly avoided in the following cases:

- Programs that SELECT from update-*sensitive* tables (when *the most up-to-date* values must be obtained from tables that are or may be updated concurrently to retrievals from them). Note that as the tables to be handled by QSEL must be explicitly defined by the administrator or the programmer, this is not really a problem.
- Programs that switch between ORACLE users (i.e., CONNECT as one ORACLE user and then CONNECT to another). Note that this is not very common.
- Programs that refer to V$ or X$ tables, directly (in a SELECT statement or any of its subqueries) or indirectly (in any view referred to in a SELECT statement or any of its subqueries).
- Programs that refer (in a SELECT statement or any of its subqueries) to a view that contains a reference to the SYSDATE built in function or to the CURRVAL or NEXTVAL pseudo columns.
- Do not use QSEL with queries that have a timestamp as a parameter (for example checking if a row is effective using the current date and time in the where clause). Each statement will definitely have different parameters, and thus will never be cached! Avoid caching such queries (the specific table combination and order can be left out of the defined table combinations) or use only date in the parameter value (so caching can be effective at least on the same day).

2.2. How to Use QSEL

2.2.1. Compilation and Linkage with Pro*C

An executable program that uses QSEL is generated from a Pro*C source program by using the *make* utility to perform the following:

1. Preprocess the source program by the QSEL preprocessor for C, *qselppoc*.
2. Precompile the QSEL preprocessor's output by the Pro*C precompiler.
3. Compile the output of the Pro*C precompiler by the C compiler, specifying the QSEL *include* directory.
4. Link the object with the QSEL *shared library*.

The above steps are performed by entering a command such as (assuming Oracle 8.x):

$ make -f /users/qsel/demo/proc80.mk userpgm

For the actual *make* file name and the required parameters, refer to your administrator.

2.2.2. Run-Time Options

A few parameters affecting the behavior of QSEL can be specified:

- Globally, for all processes, via the QSEL *global* control definitions file (/etc/qselcntl)
- For one or more specific processes, via an alternative control definitions file, the name of which is specified via an environment variable for each such specific process.
- Locally, for a specific process, via environment variables.

The values specified via the control definitions file are used as defaults, most of which can be overridden via environment variables.

2.2.3. Control Definitions Files

The default *global* control definitions file is /etc/qselcntl. An alternative control definitions file name can be specified via the QSELCTDF environment variable. In that case, if the specified name is different from the *global* control definitions file, the latter is not accessed.

A control definitions file comprises of control records, each specifying a keyword, an equals sign (=) and a value for the keyword, terminated with a new-line. No white spaces are allowed before the keyword or on either side of the equals sign. Keywords must be specified in upper-case. Comment records, starting with an asterisk (*), are allowed before, between or after control records.

The same keyword can be specified on more than one record. For all keywords except for TBNM, the *last* specification is taken. For TBNM, all specifications are taken.

The keywords are:

- **SVLV**: The minimum severity level of run-time messages to be issued by QSEL. Specify a number between 0 to 6. For more information, see "Controlling Run-Time Messages" section.
  Default: SVLV=2

- **DSAB**: Disable (Y/y) or enable (N/n) QSEL. Any value not starting with N/n will disable QSEL.
  Default: DSAB=N

- **AVLN**: The average sum of lengths of the host variables in the participating SELECT statements. Specify the number of bytes (between 1 and 3000), optionally with a K/k suffix (for Kilobytes). It is recommended that this value be obtained from The QSEL run-time statistics (for more information, see "

Run-Time Statistics" section).
Default: AVLN=10

- **MXSG**: The maximum amount of storage for use by QSEL. Specify the number of bytes (at least 1024), optionally with a K/k/M/m suffix (for Kilobytes/Megabytes). Note that too low value may degrade QSEL performance. Under the current version of QSEL, when the specified value exceeds QSEL requirements, the extra storage is not used. Note also that the amount of storage actually used by QSEL can be obtained from the QSEL run-time statistics (for more information, see "

Run-Time Statistics" section).
Default: MXSG=1M

- **SUPPRESS_LOG**: This variable allows the user to choose whether he/she wants to see licensing messages such as "no valid license exists", or "Quick-SELECT license will expire in 60 days." A value of Y/y for this variable means that these messages will not be displayed, N/n means they will. If no value is specified, the messages will be displayed (if the value of SVLV is low enough).
  Default: 'N'

- **TBNM**: A table name, or a comma-separated list of table names for a JOIN combination, to be handled by QSEL. Table name aliases, separated by spaces from the corresponding table names, are allowed (but are not used in any way). Spaces are allowed before and after commas. The syntax is similar to that of the FROM clause of a compiled SQL SELECT statement. However, under the current version of QSEL, table or view names must be simple identifiers (schemas and database links are not allowed). For QSEL to handle JOINs, all possible combinations of table names must be specified via TBNM records. For example, when all JOINs between tables *tab1* and *tab2* are to be handled by QSEL, both "TBNM=*tab1,tab2*" and "TBNM=*tab2,tab1*" have to be specified. JOINs can be specified for tables that are not to be handled separately by QSEL, and vice versa. When no table is specified, QSEL disables itself.
  Default: None

- **SUBQ**: This variable allows the user to configure where QSEL will search for table names in SELECT statements. Specifying (Y/y) will cause QSEL to parse for table names in all FROM clauses in the statement (i.e. in subqueries, union, etc.). Specifying (N/n) will cause QSEL to parse for table names only in the 1st FROM clause. QSEL will accept the entire statement only when all table names parsed are well defined in appropriate TBNM parameters.
  Default: 'N'

- **AUST**: This variable allows the user to configure QSEL whether to produce automatic statistics report when application process exits (by specifying Y/n value). This saves the application programmer the need to call the **qselst()** function in order to produce the report. For more information about QSEL statistics, see "

Run-Time Statistics" section).
Default: 'N'

Example:

```
*
* Quick SELECT sample control file
*
SVLV=1
DSAB=N
MXSG=1M
AVLN=20
SUPPRESS_LOG=N
TBNM=qselemp
TBNM=emp
TBNM=dept
TBNM=emp, dept
TBNM=dept, emp
*
* The following lines defines a JOIN between tables that are
* not to be handled separately, and specify aliases, which
* are ignored
*
TBNM=tab1 alias1, tab2 alias2
TBNM=tab2 ,  tab1  alias1
*
* The following SUBQ parameter specifies that QSEL will parse table names in all FROM clauses
*
SUBQ=Y
*
* The following will cause QSEL to produce an automatic statistical report upon process exit
*
AUST=Y
```

2.2.4. Environment Variables

Environment variables can be used for overriding the default run time options and/or those specified in the control definitions file, and for specifying an alternative control file name. The environment variable names are the same as the corresponding control definition record keywords, prefixed by **QSEL**. The valid values for the environment variables are the same as for the corresponding control definition record keywords. The TBNM control definition record keyword has no corresponding environment variable (all table names and JOIN combinations must be specified via a control definitions file). The QSELCTDF environment variable has no corresponding control definition record keyword. Environment variables specifying a null value are ignored.

- **QSELSVLV**: The minimum severity level of run-time messages to be issued by QSEL. Equivalent to the **SVLV** control definitions record keyword.

- **QSELDSAB**: Disable or enable QSEL. Equivalent to the **DSAB** control definitions record keyword.

- **QSELAVLN**: The average sum of lengths of the host variables in the participating SELECT statements. Equivalent to the **AVLN** control definitions record keyword.

- **QSELMXSG**: The maximum amount of storage for use by QSEL. Equivalent to the **MXSG** control definitions record keyword.

- **QSELCTDF**: The name of an alternative QSEL control definitions file. If the specified name is different from the *global* control definitions file, the latter is not accessed. Values from the alternative control definitions file are overridden by environment variables the same way as for the *global* control definitions file.
  Default: /etc/qselcntl (QSEL global control definitions file).

- **QSELLOGO**: Display Quick SELECT logo with version information. Values can be Y/y/N/n. Default value is Y. Use this environment variable to force Quick SELECT not to display its logo. There is not equivalent parameter in the control definitions file.

- **QSEL_SUPPRESS_LOG**: suppress or display license messages. Equivalent to the **SUPPRESS_LOG** control definitions record keyword.

- **QSELSUBQ**: sets the scope of table name parsing. Equivalent to the **SUBQ** control definitions record keyword.

- **QSELAUST**: determines whether automatic statistical report will be produced upon process exit. Equivalent to the **AUST** control definitions record keyword.

- **QSELDBG**: determines the type of debug messages that will be printed to application log file. There is no equivalent control definitions record keyword. The value of this environment variable can be one or more of the following comma separated values:
  - **funccall** will print a message on each function enter and exit.
  - **flow** will print messages on major QSEL activities.
  - **license** will print messages on license validation process.
  - **parse** will print messages on the parsing phase of the SELECT statement.
  - **hostvar** will print messages on the attributes and handling of host variables.
  - **cache** will print messages on cache activities.
  - **dump** will print messages related to various Pro*C structures that are analyzed by QSEL.
  - **all** will print all above kind of messages.

  By default, no debug information will be produced.

- **QSELORLB**: sets an alternate Oracle-Client shared library to be used. Usually, you do not need this parameter. However, in rear cases, where there is no symbolic link to Oracle-Client shared-library named libclntsh.so – then you must set the exact path and lib name in this environment variable.

2.2.5. Controlling Run-Time Messages

The minimum severity level of run-time messages defines which messages will be issued. The default is 2, which can be overridden via the QSELSVLV environment variable or the SVLV control definitions record. Valid values are :

- 0 - Informational or higher level messages.
- 1 - Warning or higher level messages.
- 2 - Error or higher level messages.
- 3 - Severe-Error or higher level messages.
- 4 - Fatal-Error or higher level messages.
- 5 - Bug messages only.
- 6 - No messages issued.

Note that when an invalid value is specified, a message *is* issued to that effect and the default remains in effect.
For more information on QSEL messages, refer to "QSEL Messages" section.

In addition to run-time messages, QSEL can also issue debug messages. See the QSEL environment variable **QSELDBG** for description how to produce debug messages.

2.2.6. Run-Time Statistics

QSEL run-time statistics can be provided for monitoring and evaluating QSEL operation. Some of the values obtained from the statistics can be used for QSEL tuning. For example, the actual average item length value should be used in order to determine the value of the AVLN (or QSELAVLN) parameter.

There are two ways to produce the report:

- By Application: The application source code **can #include "qselst.h"** in the Pro*C program. The report will be produced at that time.
- By Configuration: The application operator can configure QSEL to procude the report automatically at the application exit. See section "Control Definitions Files" and section "Environment Variables" for more details on this issue.

As a result, QSEL run-time statistics will be issued to *stdout* at the end of the program. Following is a sample of QSEL statistics:

```
    Quick-SELECT (QSEL) statistics:
    Configuration:
(01)   Max storage for cache              :  1048576 bytes
(02)   Average cache entry length         :       20 bytes
(03)Number of SQL Calls                   :     1007
(04)   Non SELECT (issued to database)    :        5/     1007 (  0.5%)
(05)   SELECTs                           :     1002/     1007 ( 99.5%)
(06)     Retrieved from cache             :      501/     1002 ( 50.0%)
(07)     Retrieved from database          :      501/     1002 ( 50.0%)
(08)       Not in cache                   :      501/      501 (100.0%)
(09)       Cache disabled                 :        0/      501 (  0.0%)
(10)       No valid license               :        0/      501 (  0.0%)
(11)       Table(s) not in config         :        0/      501 (  0.0%)
(12)       QSEL Configuration Error       :        0/      501 (  0.0%)
(13)       Cache Error                    :        0/      501 (  0.0%)
(14)       Host var Error                 :        0/      501 (  0.0%)
(15)       Unsupported SQL function       :        0/      501 (  0.0%)
(16)       Unexpected SQLCODE             :        0/      501 (  0.0%)
(17)       Internal Error                 :        0/      501 (  0.0%)
(18)Cache Activity:
(19)   Actual average cache entry length  :        6 bytes
(20)   Actual storage used for cache      :   115064/  1048576 ( 11.0%)
(21)   Optimal max storage                :  1048576/  1048576 (100.0%)
(22)   Cache refresh requests             :        0
(23)   Cache elements (current)           :      501
(24)   Cache total inserts                :      501
(25)   Cache total deletes                :        0/      501 (  0.0%)
(26)     Due to no space                  :        0/      501 (  0.0%)
(27)     Due to refresh requests          :        0/      501 (  0.0%)
(28)   Avg malloc() len                   :       32
(29)   Hash table:
(30)     Max entries                      :    24731
(31)     Used entries                     :      501/    24731 (  2.0%)
(32)     Duplicate keys:
(33)       inserted                       :        0/      501 (  0.0%)
(34)       Avg duplicate search path len  :     1.00
(35)       Max duplicate search path len  :        1
```

Report explanation:

- *Lines 01-02* show the configuration values of max storage and average entry length
- *Lines 03-17* show the SQL activity in the process:
    - *Line 03* shows the total number of SQL calls made by the application (only by modules that were compiled with QSEL).

- *Line 04* shows number of non-SELECT calls made by the application and its percentage among total of SQL calls. These calls were handled by Oracle database.
- *Line 05* shows number of SELECT calls made by the application and its percentage among total of SQL calls.
- *Line 06* shows number of SELECT calls that were fetched from QSEL cache and its percentage among total number of SELECT calls.
- *Line 07* shows number of SELECT calls that were fetched from Oracle database and its percentage among total number of SELECT calls.
- *Lines 08-17* show summary data about reasons why SELECT calls were not handled by QSEL and its percentage among all SELECT calls that were fetched from Oracle database:
  - *Line 08* shows number of times data was not found in cache.
  - *Line 09* shows number of times QSEL was disabled.
  - *Line 10* shows number of times an invalid QSEL license was detected.
  - *Line 11* shows number of times that update-sensitive tables were used in the SELECT.
  - *Line 12* shows number of times caused by wrong QSEL configuration.
  - *Line 13* shows number of times a cache error occurred.
  - *Line 14* shows number of times an improper host variable definition was used.
  - *Line 15* shows number of times an unsupported SQL function or attribute was used.
  - *Line 16* shows number of times an unexpected SQLCODE returned from Oracle.
  - *Line 17* shows number of times an internal error in QSEL occurred. Important: QSEL internal errors might occur due to bug in QSEL or bug in Pro*C.

- *Lines 18-35* show the QSEL cache activity:
  - *Line 19* shows the average length of cached entry.
  - *Line 20* shows the actual storage used for cache.
  - *Line 21* shows the optimal max storage definition calculated by QSEL and its percentage relative to the configured max storage. When actual storage used is less or equal than configured value, the optimal max storage is as the configured. Only when the actual storage is greater than configured then the optimal value gets beyond 100%, which implies that user should consider extending the max storage configuration parameter to avoid cache-deletes and improve performance.
  - *Line 22* shows the number of time user application requested a cache refresh (e.g. clean the cache and start caching from scratch).
  - *Line 23* shows the number of cached entries currently stored in the cache.
  - *Line 24* shows the total number of entries inserted into the cache.
  - *Line 25* shows the total number of entries deleted from the cache and its percentage among total number of cache-inserts. The lower this percentage gets – the best performance is achieved.
  - *Line 26* shows the total number of entries deleted from the cache because there was not enough space in the cache to store new data. Also shown is the percentage among total number of cache-inserts.
  - *Line 27* shows the total number of entries deleted from the cache because of cache-refresh requests. Also shown is the percentage among total number of cache-inserts.
  - *Line 28* shows the average length (in bytes) of each dynamic memory allocation made by cache mechanism.
  - *Lines* 29-35 shows statistics on the hash mechanism which is the core of the cache mechanism:
    - *Line 30* shows the size (in entries) of the hash-table defined for this run.
    - *Line 31* shows how many hash-table entries were actually used and its percentage among the size of the hash-table.

- *Line 33* shows how many duplicate hash keys were inserted into the hash-table. A duplicate key means that same hash-table entry holds more than one cached data. The lower this value gets – the better performance is achieved.
- *Line 34* shows the average number of duplicates searched till the right-cached data fetched from cache. When this value is 1 – the search is considered most efficient.
- *Line 35* shows the maximum number of duplicates searched till the right-cached data fetched from cache. When this value is 1 – the search is considered most efficient.

2.2.7. QSEL Cache

2.2.7.1. Overview

To achieve performance improvement, Quick-SELECT caches data retrieved from Oracle in a cache mechanism. When a SELECT statement is issued by application program, Quick-SELECT first looks for the data in the cache. If data is already in cache, Quick-SELECT retrieves the cached data to calling application. Otherwise, Quick-SELECT delegates SQL statement request to Oracle and only then caches the data retrieved from Oracle allowing subsequent request for same data to be retrieved cacheable.
Since Quick-SELECT runs under the calling application process, one cache is defined per process.

2.2.7.2. Caching Unit

The unit of caching includes all values of input host-variables (those appearing in the WHERE clause) and all values of output host-variables (those appearing in the INTO clause) associated with each invocation of a SELECT statement.

2.2.7.3. Structure & Content

Quick-SELECT cache is Hash (key/data) based mechanism.

Entry Key:
The key is a concatenation of the following fields:

- SQL Stmt ID – This is the unique ID given by Quick-SELECT to each accepted SELECT statement.
- Input Host Variables – All values of host variables appearing in the WHERE clause of the statement including.

Entry Data:
The data is a concatenation of all output host variables (including null indicators, length fields where needed) appearing in the INTO clause.

2.2.7.4. Cache Management

For each accepted statement with no previous errors, Quick-SELECT will attempt to cache the data retrieved from Oracle.

- When storage limitation allows and Hash table is not full, Quick-SELECT will allocate new key/data buffers and update Hash table with the new cache entry.
- When cache is full, Quick-SELECT will first delete as many cache entries as needed in order to free enough space for new cache entry to be added. The cache entries are deleted in Least Recently Used order, meaning that old records are deleted first. The impact of deleting an entry from cache is that if it is required later, Oracle will be called first to get the data before it can be re-cached by Quick-SELECT.
- When cache memory is too small to hold even one record, Quick-SELECT will produce an error message and stop handling that SQL statement.

2.2.7.5. Cache vs. QSEL Control Parameters

The following QSEL Control parameters controls the cache size and behavior:

MXSG (or QSELMXSG environment variable):
Defines the maximum storage Quick-SELECT is allowed to use for both the cache keys and data.

AVLN (or QSELAVLN environment variable):
Defines the expected average length of the sum of input host-variables in the key

Based on those parameters values, Quick-SELECT calculates the size of cache Hash table and allocates memory for it. Memory for key/data is allocated only when required.

The values of the AVLN & MXSG parameters have major impact on the efficiency of Quick-SELECT caching mechanism resulting with impact on application performance.

It is recommended that each process will use different values of those parameters and not use a global definition that applies to many other processes running with Quick-SELECT.

2.2.7.6. Determining AVLN & MXSG Values

The following steps suggest a way to determine the appropriate parameter values for your application. Be aware to the implications of changing those parameters value (see "Cache Performance Considerations" section).

1. At first, try to estimate the values of those parameters.
   - The AVLN value can be estimated according to the nature of the SELECT statements in your application. If you can't estimate it, leave that parameter and let Quick-SELECT use its default value (i.e. 10 bytes).
   - The MXSG value can be set according to your requirements or environmental limits. If you can't estimate it, let Quick-SELECT use its default value (i.e. 1MB).

2. #include Quick-SELECT Statistics routine just before your application ends or configure QSEL with the **QSELAUST** parameter in order to produce the statistical report.

3. Compile your application programs (with QSEL).

4. Run your application program.

5. Examine Quick-SELECT statistics report:
   - The "`Actual average cache entry length`" field shows the real average of item length. Use this value in the AVLN parameter.
   - The "`Actual storage used for cache`" field shows how much storage was actually used by Quick-SELECT and the percentage of it related to the MXSG value defined. If the percentage is very low you can decrease MXSG value. If the percentage is high, consider increasing the MXSG value.

6. If you are satisfied with the results – skip to step 9.

7. Update the Quick-SELECT control parameters (or environment variables) according to your findings.

8. Return to step 4.

9. Populate the parameters values to be permanent and available for your application for future runs.

2.2.7.7. Cache Performance Considerations

The following should be considered when setting values to those parameters.

1. When AVLN decreases or MXSG increases:
- Less chance for Hash Duplicates
- Hash efficiency increases
- Less need to delete cache entries
- Memory wasted for larger Hash table

2. When AVLN increases or MXSG decreases:
- More chance for Hash Duplicates
- Hash efficiency decreases
- More need to delete cache entries
- Less memory allocated for Hash table

3. When Quick-SELECT statistics show that memory utilization percentage is high, it may imply that cache parameters tuning is required since there might be shortage in cache memory.

4. Cache deletes might cause performance degradation.

2.2.7.8. Cache Refresh

An application program can decide that data cached by QSEL is not needed any more and can request QSEL to refresh its cache memory.
Cache refreshing is performed over all data currently in QSEL cache. Current QSEL version does not support any conditional refresh (via parameters).

QSEL cache refresh is accomplished by a new routine, called *qselref().* This routine deletes ALL elements in QSEL's cache buffers. Thus, calling *qselref()* clears QSEL's cache buffers. Consequently, the SQL SELECTs which follow a *qselref()* call will result in a reference to Oracle's data base, and not to QSEL's cache.

In order to clear QSEL's cache buffers, code the following line:
In Pro*C:

```
qselref();
```

2.3. QSEL Messages

QSEL run-time messages are issued on standard error file (*stderr).*
QSEL run-time messages are grouped into categories as follows:

- 0 - Informational or higher level messages.
- 1 - Warning or higher level messages.
- 2 - Error or higher level messages.
- 3 - Severe-Error or higher level messages.
- 4 - Fatal-Error or higher level messages.
- 5 - Bug messages only.
- 6 - No messages issued.

Severity level	**Message Prefix**	**Type of messages**
(0) Informational	I	Describe which SQL statements have been accepted by QSEL
(1) Warning	W	Describe the SQL statements that have been rejected by QSEL, detailing the reason
(2) Error & (3) severe	E	Issued for control definitions file errors or problems that prevent QSEL processing the specific SQL statement.
(4) Fatal	F	Issued for unintentional QSEL termination/disabling.
(5) Bug	B	Describe internal QSEL failures caught while QSEL checks its operation

Any QSEL message has the following format:
<*msg-prefix*>-<*QSEL module*>: <*error description text*>

<*msg-prefix*> Is the message prefix as described in the table above.
<*QSEL module*> Is the QSEL module detected the error.
<*error description text*> Is the error description text.

QSEL Logo message (Version information) is issued once per running process and only when one of the following is true:
- QSEL successfully cached the output of at least one SELECT statement
- Some QSEL messages issued (like warnings/errors).

Note that even in the case of fatal errors and caught bugs, the user program continues to work normally, only without the performance improvements.

For a complete list of messages please see section "Appendix: QSEL Run-time Messages".

Be aware that for problem determination purposes, it might be needed to use lower severity level so that more messages will be logged and help investigate the problem. In addition, you can use the QSEL debug option by configuring the **QSELDBG** environment variable.

3. Quick SELECT Licensing

3.1. General Information

3.1.1. Overview

Quick SELECT requires a valid license in order to function properly. The lack of valid license will cause Quick SELECT to route SQL SELECT statements directly to Oracle without caching any data.

Therefore, after installing Quick SELECT, the following activities should be performed by the administrator:

- Running Quick SELECT Registration Program. This program creates a Registration File and a temporary License File that is valid for 60 days.
- Issue a License Request via Quick SELECT Registration web site. This request will include the Registration File.
- Upon receiving a valid license file from Log-On, it should be installed into Quick SELECT directory.

3.1.2. Terminology

The following software components are used regarding licenses:

- **Runtime Library** – The regular Quick SELECT shared object.
- **Registration Program** – A new Quick SELECT program that creates a Registration File
- **License Server** – A new Quick SELECT program that is allowed to create licenses for hosts in site
- **License Control Program** – A new Quick SELECT program that helps the administrator to maintain licenses
- **Registration Web Site** – A new web site that should be used to requests licenses from Log-On Software

The following terms are used regarding licenses:

- **Long-term License** – A license that is created by Log-On Software for user based on his support contract
- **Temporary License** – A license that is created by the Registration Program and is valid for 60 days
- **Emergency License** – A license that is provided via Quick SELECT Registration Web Site and is valid for 10 days.
- **Grace Days** – A period of time (in days) after license has expire during which Quick SELECT will still function properly.

3.1.3. When is a License required?

As from version 1.6.02, Quick SELECT is packaged and delivered in two tar files:

- **DEVELOPMENT tar file –** Contains a Development Runtime version of Quick SELECT. This version does not need a license but it does not cache anything. All SELECT statements are routed directly to Oracle.
- **PRODUCTION tar file –** Contains a Production Runtime version of Quick SELECT. This version requires a valid license to be installed and is fully functioning when one exists.

3.1.4. License Scope

3.1.4.1. Site License

Site license means that all hosts in the site share one license.
To obtain a site license, Quick SELECT License Server Program should registered once via Web site to run on certain host in site.
The provided license allows that License Server to run on that host and create licenses for all hosts in site that need to run Quick SELECT Runtime Library.

It is possible to have multiple License Servers in a site.

3.1.4.2. Stand Alone License

Standalone license means that a license should be provided for a Runtime Library running on host that is not connected to any License Server in site.
To obtain a Standalone license, Quick SELECT Runtime Library should registered via Web site to run on certain host in site.
The provided license will allow Quick SELECT Runtime Library to run only on that host.

3.1.5. License Validation

In any configuration, no matter if license was obtained from Web site or from License Server running in the site, each copy of Quick SELECT product in the site must have a valid license before it can operate.

When Quick SELECT Runtime Library encounters the first SELECT statement in an application process, it first validates the license installed in the local Quick SELECT directories. During the license validation, there are no interactions with the License Server.

The license validation procedure checks that the current host is authorized to run this version of Quick SELECT and that expiration date has not been reached. For host identification, the host IP address specified in the License File must match one of the IP addresses configured for the host.

If the local license is valid, Quick SELECT will continue to work normally and cache the data for the entire application process.
If the local license is valid but its expiration date is close, Quick SELECT will display a warning message and continue to work normally.
If the local license is not valid, Quick SELECT will display an error message and all SELECT statements will be routed directly to Oracle.

3.2. New Files in Quick SELECT Directory

3.2.1. New files in the bin directory

Several new executable files were added to the bin directory (in PRODUCTION version only):
qsellicd – Quick SELECT License Server Daemon
qsellicc – Quick SELECT License Control Program
qselinfo – Print Quick SELECT Version Information.
qselreg – Create Quick SELECT Registration file

3.2.2. New directories under QSEL Home directory

Two new directories were added to the PRODUCTION version of Quick SELECT:

- **License** – contains the Registration files created by Quick SELECT Registration Program and the License files.
- **Log** – contains the Quick SELECT License Server's log file

3.3. Registration Program (qselreg)

3.3.1. General

The Registration Program (**qselreg**) must be executed after extracting Quick SELECT tar file. It will create the following files:

- Appropriate Registration file to be used while requesting a license in Quick SELECT Registration Web site.
- A License File for the registered product.

During its execution, the Registration Program will try to connect to a Quick SELECT License Server running in the site in order to receive a license key.
If such a server exists, the server will create a license key which will be written to the License File.

If no License Server is active, the Registration Program will create a temporary License file with a valid license for 60 days.

All Registration & License Files are created in the **license** directory.

3.3.2. Invocation

Run this program from the shell prompt.
If you want to register Quick SELECT License Server type: **qselreg -s**
If you want to register Quick SELECT Runtime library type: **qselreg -q**
If you run the program with no parameters, the program will guide you through its options.

3.4. License Server (qsellicd)

This is the Quick SELECT component that is allows to use Site License.

Before using the License Server, A Registration file must be created and a valid license must be obtained from the Quick SELECT Registration Web site.

It is recommended to first install a License Server in site, obtain a license for it and then register other Quick SELECT copies on other hosts in site.

Only one License Server can be active on single host at a time.
It is possible to have multiple hosts running License Server each.

It is important to understand that the connections to the License Server are made only when registering new copy of Quick SELECT product. During user application execution there is no use of the License Server.

The License Server is running as a daemon.

The startup/shutdown of the License Server are described in the next section.

3.5. License Control Program (qsellicc)

3.5.1. General

The License Control Program is the interface between the administrator and the Quick SELECT License layer. It allows to perform several actions described below regarding licenses.

3.5.2. Invocation

The License Control Program can be activated using one of the following forms:

qsellicc –s Start License Server
qsellicc –q Stop License Server
qsellicc –I Install new license
qsellicc –u Update existing license
qsellicc –p Print license information
qsellicc –d Dump license file content
qsellicc –w Print list of active License Servers
qsellicc –h Print usage

See next sections for further details.

3.5.3. Start License Server

Use the command: **qsellicc –s**
to launch a License Server on current host.

The License Server log file is written to the **log** directory under **qselhome**.

3.5.4. Stop License Server

Use the command: **qsellicc –q**
to stop the License Server running on current host.

You cannot issue this command to stop a server running on other host.

3.5.5. Install New License File

Use the command: **qsellicc –i file**
to install new license file named **file** into current host.

Installing License File is required whenever you receive a Site license file or Stand alone license file from Log-On Software.
Upon successful install, the new license will override the old license if existed.

Note: It is not required to have an active License Server when issuing this command.

3.5.6. Update Existing License File

Use the command: **qsellicc –u server | qsel**
to update/refresh the currently installed license on current host.

Updating a license is required after you have installed a new Site License and you want to update the license of all Quick SELECT copies in site. Such an activity requires that the administrator will issue this command from each host in site that runs Quick SELECT and needs a license update.

The license update process will connect to the active License Server in the site in order to get an updated license.

If you want to update the license of a License Server, use the parameter **server**. Otherwise, use the parameter **qsel**.

Note: It is required to have an active License Server when issuing this command.

3.5.7. Print License Information

Use the command: **qsellicc –p server | qsel**
to print license information.

If you want to print information about the license of the active License Server, use the **server** parameter (verify that there is an active server first).

If you want to print information about the license of the Runtime library on current host, use the **qsel** parameter.

Following is an example of a license information report:

```
Internal License Version        :  1
Quick SELECT component licensed : Quick SELECT Runtime Library (*1)
Created By                      : Quick SELECT Runtime Library Installer (*2)
Allowed to run on host (ID)     : salsa (199.203.211.119) (*3)
License expiration criteria     : Quick SELECT version and Expiration Date (*4)
Quick SELECT version licensed   : 1.9.02 (*5)
Expiration Date                 : 2013/11/26 (YYYY/MM/DD) (*6)
Grace Period Allowed (in days)  : 60 (*7)
Can create license for other server : No (*8)
```

Where:

(*1) Shows the name of the Quick SELECT component that the license applies to. Can be "Quick SELECT Runtime Library" or "Quick SELECT License Server".

(*2) Shows who created this license

(*3) Shows the host name and IP address to which the license applies.

(*4) Shows the condition that will be used to verify whether license is valid or expired. Condition can be one of the followings: Quick SELECT version / Expiration Date / Quick SELECT version and Expiration Date.

(*5) Shows the Quick SELECT version number applies to this license (this information is shown only when condition includes Quick SELECT Version)

(*6) Shows the date when license expires (this information is shown only when condition includes Expiration Date).

(*7) Shows number of grace days allowed after expiration date has arrived (this information is shown only when condition includes Expiration Date).

(*8) Shows whether this license allows its holder to create license for others (this information is relevant only for license of a License Server)

3.5.8. Dump License File Content

Use the command: **qsellicc –d file**
to print the content of a specific license file.

You can use this command to display the content of any license file named **file**.

The displayed output format is the same as in the –p option.

3.5.9. Print List of Active License Servers

Use the command: **qsellicc –w**
to print a list of all active License Servers in the site.

3.6. Quick SELECT Registration Web Site

3.6.1. General

The Quick SELECT Registration Web site is intended to serve Quick SELECT customers' requests regarding Quick SELECT licenses. The site's address is: www.qsel4oracle.com

A license request might be handled automatically by the web server or be queued for later manual handling by Log-On representative.

In any case, the response to the customer's request will be sent by email.

The following sections will describe how to fill the request form.

3.6.2. Specifying the License Type Requested

First, you should specify what is that you request to get. You can select one of the following options:

- **An emergency license** – If for some reason you don't have a valid license installed in your site, you can request an emergency license that will help you get through. The license you receive is valid for 10 days. It is expected that you will renew your license within that period of time.
- **A long term license** – Select this option if you want to receive a license that is valid for long period of time.

- **Copy of existing license** – Select this option if you already got a license but you want a copy of it (e.g. your license file has been damaged).

3.6.3. Specifying Registration Information

You must supply Quick SELECT registration information with each license request.
Specifying Registration information can be done in one of the following methods:

- **File Upload –** This option lets you upload the registration file created when you installed Quick SELECT. The required information will be extracted from that file.
- **Manual –** This option lets you specify manually the registration information required to fill your request.

3.6.4. Specifying Personal Details

Finally, you must also supply your personal details so we can contact you later if there is a need to.

3.6.5. Processing Your Request

After filling all details mentioned above, press the SUBMIT button so that your request can be sent and handled.

If your request is for an emergency license or copy of existing license then your request will be handled automatically in short time and a response email will be sent to the email address you specified in the request. That email response will contain the desired license file.

If your request is for a long term license then it will be saved and queued for later processing by Log-On Software representative. Please allow several days for processing your request. Note that the license you will receive eventually is limited by conditions set by Log-On Software.

The license file you receive can be installed on your host by using Quick SELECT License Control Program (**qsellicc**).

4. Administrator's Guide

4.1. Before Installing Quick SELECT

As from Quick SELECT version 1.6.02, you need a license to use the product (refer to chapter "Quick SELECT Licensing" for more information.

If you are interested in a site license, it is recommended to perform the installation in the following order:

1. Install Quick SELECT PRODUCTION tar file on the host where Quick SELECT License Server will run
2. Run the Registration Program (**qselreg**) to register the License Server.
3. At this point you have a temporary license for 60 days to use the License Server.
4. Go to Quick SELECT Registration Web site and fill a License Request form for a Server license.
5. Obtain a license from Log-On Software.
6. Install the Server license you got (via **qsellicc**).
7. Launch Quick SELECT License Server (via **qsellicc**)
8. On each host where you want to use Quick SELECT:
   - Install Quick SELECT PRODUCTION tar file
   - Run the Registration Program to register Quick SELECT Runtime Library (shared object). This will automatically install the site license on current host by accessing the License Server.

Note: If you skipped steps 4 and 5 above and performed those at a later time, you will need to update the license on each host in your site that Quick SELECT is installed on.

4.2. Installing Quick SELECT

4.2.1. Terminology

Following terms are being used in the following sections:

CC	Configuration Control. Tool for managing Software configuration
Data Layer	Program(s) that access Oracle via embedded SQL.
Application	Any application that uses Quick SELECT

4.2.2. Required Privileges

Only account with super-user privileges (like root) can install Quick SELECT.

4.2.3. Disk Space Requirement

Quick SELECT installation process is very simple and requires about 1-2 MB of disk space.

4.2.4. The Installation File

You must use the appropriate installation file that suits your target host/environment.

The Quick SELECT installation file name depends on the following attributes:

- Quick SELECT version
- Operating System name and version
- Oracle Client version.
- Executable architecture (32/64 bit)
- DEVELOPMENT or PRODUCTION – Indicates whether version requires license (PRODUCTION) or not (DEVELOPMENT). The DEVELOPMENT version will not cache any data.

4.2.5. Creating Quick SELECT directory

Once the configuration has been defined the following steps must be done:

1. Make sure that there are all necessary permissions to read/write in directory **/opt**.

2. **For SUN or Digital computers**: Create the new QSEL directory under the **/opt** directory. The new QSEL directory name is usually /opt/qsel*mnr* where *mnr* is the Quick SELECT version (Major, Minor, Release). For instance: the new QSEL directory for Quick SELECT 1.4.03 is /opt/qsel*1403*).

3. **For HP computers:** On HP computers the application binaries are usually compiled without the '-Wl,+s' flags. As a result, the applications expect the shared libraries to reside at run time at the exact same paths were they resided at build time; the environment variables *SHLIB_PATH* and *LD_LIBRARY_PATH* are ignored. Therefore, if it is possible and desirable not to recompile the data layers on the HP machine the new Quick SELECT release must be installed at the same path of the previous one. So in this case you should proceed as follows:
   (a) Make sure no running process is using Quick SELECT.
   (b) Rename the current Quick SELECT directory to *.old.
   (c) Create the new Quick SELECT directory with the old Quick SELECT directory name.

4. Permit others to read & execute Quick SELECT files:
   $ chmod a+rx <Quick SELECT *qselhome* dir>

4.2.6. Extracting Quick SELECT Directory Tree

1. Copy QSEL installation file to the new Quick SELECT directory.

2. Enter into the QSEL home directory that you have just created.

3. Extract the Quick SELECT *qselhome* directory tree by issuing the following command:

$ gzip -dc <Quick SELECT tar file> | tar –xvf -

4.2.7. Content of Quick SELECT Directory Tree

The following directories are extracted from the tar file:
bin – Contains Quick SELECT executable file
demo – Contains Quick SELECT demo programs and make files
lib – Contains Quick SELECT shared object and important C header files required for user application build
license – Contains Quick SELECT Registration and License Files. This directory exists only in the PRODUCTION version.
log – Contains Quick SELECT License Server log files. This directory exists only in the PRODUCTION version.

4.2.8. Registering Quick SELECT

If you installed the PRODUCTION tar file, then you must register Quick SELECT and obtain valid key before you can use the product.
To register Quick SELECT, run the program qselreg located in the bin directory. For more information about Quick SELECT licensing please see "Quick SELECT Licensing" chapter.

4.2.9. Adjust Make File

Adjust the QSEL global control definitions file, $QSEL_HOME/demo/qselfile.ctl, specifying the table names which QSEL is to handle. For more information, see "Control Definitions Files" section. At this stage, please leave in the line specifying TBNM=qselemp, which is used for the verification of the installation later on.

Create a *make* or script file for using QSEL services from Pro*C programs. For more information, see "Appendix: Adjusting the Precompiler Makefiles" section.

According to your installation, choose the appropriate sample makefile:
Pro*C 1.5: demo/proc15.mk

Pro*C 1.6: demo/proc16.mk

Pro*C 8: demo/proc80.mk

Pro*C 9: demo/proc90_64bit.mk_OSF1 , demo/proc90_32bit.mk_SUN, demo/proc90_64bit.mk_SUN, demo/proc90_32bit.mk_HP, demo/proc90_64bit.mk_HP

Edit the QSEL sample compilation makefile, and adjust the value assigned to QSEL_HOME (in the first non-comment line) to the name you selected for it.

4.2.10. Post Installation Activities

1. Add or renew links to QSEL home directory (*qselhome*) in CC.

2. In order to work properly with Quick Select, the Data Layer programs must be recompiled with QSEL's preprocessor. It is very important that the version of Quick SELECT installation file corresponds to the OS and Oracle versions. Sometimes, when upgrading to higher version of Quick SELECT, the recompilation of the data layer is not necessary. In that case you should get specific instructions along with the installation file.

3. Create or renew the following environment variables for all user accounts and all applications' scripts:
   - $QSEL_HOME - points to the new location of the Quick SELECT *qselhome* directory.
   - $QSELCTDF - points to the Quick SELECT control file (in the CC or any other preferred location)
   - $QSELDSAB ($QSELDSAB = N means that Quick SELECT is active).
   - Make sure that directory $QSEL_HOME/lib is part of the directories list of $SHLIB_PATH, $LD_LIBRARY_PATH and $PATH.

4.2.11. Installation Verification Process

The following steps verify that the installation is successful**.**

4.2.11.1.Environment

1. Set the following environment variables:
   - $QSEL_HOME - points to the new location of qsel
   - $QSELCTDF - points to control file in $QSEL_HOME/demo/qselfile.ctl
   - $QSELDSAB - N (that is, Quick SELECT is active).
   - $QSELSVLV – 0 (that is, report all Quick SELECT messages)

2. Make sure you have access to an Oracle account with CREATE TABLE privilege.

3. Make sure that directory $QSEL_HOME/lib is part of the directories list defined by $SHLIB_PATH / $LD_LIBRARY_PATH environment variables.

4.2.11.2.Compile Quick SELECT Sample Program

Login to another user. If ORACLE security is by the operating system, the user must have a CREATE_ANY_TABLE privilege in ORACLE.

Copy the demo/qselsamp.pc (for Pro*C) program from the QSEL home directory to your current directory. For example:

$ cp /users/qsel/demo/qselsamp.pc /users/myuser

This program CREATEs a table, INSERTs one row into it, SELECTs that row 10 times, and DROPs the table.

Compile qselsamp.pc or qselsamp.pco using the adjusted *make* file. For example:

$ make -f /users/qsel/demo/proc80.mk qselsamp

4.2.11.3.Run Quick SELECT Sample Program

Set the QSEL minimum message severity level to 0.

For example, under the Bourne or Korn Shell:

$ QSELSVLV=0
$ export QSELSVLV

Or, under the C Shell:

$ setenv QSELSVLV 0

Run qselsamp, specifying a userid and a password for CONNECTing to ORACLE. If ORACLE security is *not* by the operating system, the userid must have a CREATE_ANY_TABLE privilege in ORACLE. For example:

$ qselsamp scott/tiger

A few informational and warning messages are issued, indicating which SQL statements are handled and which are not. Eventually, QSEL statistics are printed, indicating that of the 10 SELECTs issued, just one accessed the database.

Finally, the line in the global control definitions file specifying TBNM=qselemp should be removed. Note that for editing the global control definitions file, you may have to switch to a user permitted to write on the file.

4.3. Compilation *make*/script Files

Programmers should have the option to determine, for each program being compiled, whether it will use QSEL services or not. Thus, either a separate *make*/script file should be prepared for compilation/linkage using QSEL, or the same *make*/script file can have an option to use or not to use QSEL. Our recommendation is to have separate *make*/script files - retaining the existing *make*/script file(s) and creating a new one for using QSEL.

4.3.1. The QSEL Preprocessor for C

The QSEL preprocessor for C, *qselppoc*, prepares a Pro*C program for working with QSEL. It must be invoked *before* the Pro*C precompiler.
The QSEL preprocessor for C has no command-line arguments.

The preprocessor reads a Pro*C program from standard input, and writes the preprocessed program to standard output. Thus, redirections are required. Standard input has to be redirected ("<") from the original Pro*C program (usually with the ".pc" name suffix). Standard output has to be redirected (">") to a new file (usually with a ".qc" name suffix). The input to the Pro*C precompiler has to be taken from the file generated as the redirected output of *qselppoc* (the ".qc" file).

For example:

$ $QSEL_HOME/bin/qselppoc <userpgm.pc >userpgm.qc
$ $ORACLE_HOME/bin/proc iname=userpgm.qc

The standard ORACLE Pro*C makefile can be duplicated and adjusted for using QSEL. Moreover, the QSEL installation tape contains such an adjusted sample makefile.

4.3.2. The Include Directory and Shared Library

For the C compiler (*cc*) invocation, QSEL *include directory* ($QSEL_HOME/lib) must be specified (with the -I option). The QSEL *shared library* ($QSEL_HOME/lib/libqsel.so) must be specified as input to the C compilers or to the linker (*ld*), whichever is invoked for performing the linkage.

For example, compiling and linking with the C compiler:

**$** cc -I. *-I$QSEL_HOME/lib* -O -Aa -Wl,-aarchive
-L$ORACLE_HOME/lib -o userpgm userpgm.c
$QSEL_HOME/lib/libqsel.so $ORACLE_HOME/lib/libsql.a
$ORACLE_HOME/lib/osntab.o -lsqlnet -lora
$ORACLE_HOME/lib/libpls.a -lsqlnet -lnlsrtl -lcv6
-lcore -lnlsrtl -lcv6 -lcore -lcl -lm

4.3.3. Adjusting the Compilation *make* Files

The QSEL installation directory ($QSEL_HOME/demo) contains the proc15.mk, proc16.mk (hence proc*nn*.mk), proc80.mk and the procob16.mk makefiles.

The proc*nn*.mk makefile invokes the QSEL preprocessor for C, *qselppoc*, for preprocessing ".pc" programs into ".qc" programs. It then invokes the Pro*C precompiler for precompiling ".qc" programs into ".c" programs. Finally, the C compiler is invoked for generating executable programs.

The proc*nn*.mk makefiles are modified copies of the $ORACLE_HOME/proc/demo/proc.mk for Pro*C 1.5, Pro*C 1.6 and Pro*C 8 respectively. These makefiles can be used as are, but it is recommended that they be re-generated from the current ORACLE Precompiler makefiles currently in use. For details, see "Appendix: Adjusting the Precompiler Makefiles" section.

The modified proc*nn*.mk makefiles provide the support for compiling programs with QSEL. However, they can still be used for compiling programs without QSEL, by invoking a dummy command (e.g., *cat* or *more*) instead of the QSEL preprocessor.

Here is a list of all the possible invocations:

1. To compile a Pro*C program using QSEL:
   $ make -f proc*nn*.mk userpgm
2. To compile a Pro*C program without QSEL:
   $ make -f proc*nn*.mk userpgm QSELPPOC=cat

4.4. Platform and/or ORACLE Migration

Administrators should consult with Log-On in advance prior to migrating to another computer type or to another version of the operating system, ORACLE or the Precompiler.

4.5. Disabling Quick SELECT

There are a few possibilities for disabling QSEL:

1. For a *specific program*, by compiling the program without the QSEL preprocessor.
2. For a *specific process*, by setting the QSELDSAB environment variable to Y.
3. For *all processes that specify the same alternate control definitions file* (via the QSELCTDF environment variable) *but do not specify QSELDSAB=N*, by specifying DSAB=Y in that control file.
4. For *all processes specifying neither an alternate control definitions file nor QSELDSAB=N*, by specifying DSAB=Y in the *global* control file.
5. For *all processes*, by replacing the QSEL shared library with another shared library containing a dummy QSEL. Here QSEL cannot be enabled via the QSELDSAB environment variable. Two *make* files are provided in $QSEL_HOME/demo for completely disabling and enabling QSEL. These *make* files should be used by a user with a write access to the lib directory in the QSEL home directory. To disable QSEL globally, execute *make* for the qseldsab file, specifying the QSEL home directory. For example:

$ make -f /users/qsel/demo/qseldsab QSEL_HOME=/etc/qsel

To re-enable QSEL globally, execute *make* for the qselenab file, specifying the QSEL home directory:

$ make -f /users/qsel/demo/qselenab QSEL_HOME=/etc/qsel

4.6. Displaying Quick SELECT Version

A new utility program (**qselinfo**) was added to the bin directory. Run this program (no parameters required) to print Quick SELECT version information.

5. Appendix: Release Notes

This chapter will detail (where possible) the changes made in Quick SELECT vesions.

5.1. Version 1.1

Release 1.1 (July 1995)

- Quick SELECT now supports host arrays.
- Quick SELECT now supports performance improvements for "no data found" conditions. Note that this implies that concurrent additions to the accessed table are not usually recognized.

Release 1.12 (November 1996)

- Quick SELECT now supports Pro*COBOL.
- Quick SELECT now supports cache refresh, enabling the user to clear Quick SELECT cache buffers. This new functionality is accomplished by a new routine, called **qselref()**. For more information refer to "QSEL Cache" section.

5.2. Version 1.3

Release 1.3.01 (March 2000)

Bug Fixed: Quick SELECT did not expect sqlety=256. The bug caused the following message to be produced: "`B-qselcex: unexpected sqlety value (256)`"

Release 1.3.02 (May 2000)

- Now upon initialization Quick SELECT writes a logo to the standard output. For example:

```
I-qselcex: Quick SELECT version 1.3.02 for ORACLE 8.0.5 under OSF1 V4.0 (alpha).
(c) Copyright 1994-2000 Log-On Software. All rights reserved.
```

  When the level is 0 Quick SELECT writes additional building information.

- Whenever applicable, the application program file name and line number are printed as part of each diagnostics message
- Fixed the bug that generated the following message:

```
B-qselcex: sqlstm.cud is NULL or sqlstm.offset is invalid
```

Release 1.3.03 (June 2000)

- Fixed the bug that caused Quick SELECT to abort with a core dump under HP machines.
- Removed the '-g' flag in Quick SELECT make file, resulting in a much smaller Quick SELECT shared library.
- Added the '+O2' optimization flag to the Quick SELECT make file.

Release 1.3.04 (July 2000)

Simplified the process of porting Quick SELECT to different platforms.

Release 1.3.05 (August 2000)

- All Quick SELECT messages are printed on the standard error rather than the standard output.
- Fixed and refined the validation of PRO*C 8 variables.
- Failure of the validation of the PRO*C 8 variables is now reported as a BUG (B-) rather than a warning (W-).

- Failure of the validation of the PRO*C 8 variabes is now reported as

```
"B-qselcex: Validation of Pro*C 8.x variables failed (-2)."
```

rather than

```
"B-qselcex: One of these vars : Sqphss, Sqpins, Sqpadto, Sqptdso Have a Value
Diffrent than 0."
```

Release 1.3.06 (September 2000)

- Extended Pro*C 8 variables validations
- Improve Quick SELECT porting to different platforms

Release 1.3.07 (February 2001)

This release fixes the BUG that showed up while SELECTing into host arrays and was generating error messages such as the following example:

```
"E-qselcex: SQL statement in file "xxxxxx.pc" line 441 will NOT be handled -
actual length (8224) of host variable [1][2] exceeds its maximum length (20)"
```

The bug was caused by improper analyze of some Oracle variables.

Release 1.3.09 (April 2001)

- Fixed bug consisted of Quick Select overrunning its dynamically-allocated buffer by exactly one byte in the very rare case where the SQL SELECT data was exactly 256 bytes long.

- Clarified several messages by appending to them ", ID = <id>" where <id> is the statement ID assigned by Quick SELECT. Here is an example:

```
"I-qselcex: SQL statement in file "xxxx.pc" line 4789 accepted, ID = 2"
```

Release 1.3.10 (July 2001)

Fixed the error that sometimes occurred when the host computer was heavily loaded. When the error occurred, Quick SELECT was disabling itself and generating the following message:

```
"E-qselcex: read(2) from pipe failed: Interrupted system call."
```

Release 1.3.10a (August 2001)

Disabled the Oracle server version check. When this check failed it used to disable Quick SELECT and generate a message such as the following:

```
"E-qselcex: This build does not support Oracle 8.1.7 !"
```

That error occurred only when Oracle client version did not match Oracle server version.

Release 1.3.11 (October 2001)

Write the Quick SELECT logo to the "/tmp/qsel.log" file to allow alternate source for retrieving Quick SELECT version information during problem determination.

Fixed the bug that generated messages similar to the following:

```
D-qselcex: PC80_check: No. of vars ...
D-qselcex: PC80_VARS: ix =0: v1S =34359738376, v1L = 8 ...
...
B-qselcex: SQL statement in file "dlr.qc" line 132 will NOT be handled -
Validation of Pro*C 8.x variables failed (-2).
```

5.3. Version 1.4

Release 1.4.02 (November 2001)

- Quick SELECT Logo is displayed if Quick SELECT actually caches or if a warning or error message needs to be displayed. Logo is printed once per process.
- Quick SELECT does not generate defunc children anymore.

Release 1.4.02a (November 2001)
The logo information has been squeezed to one line.

Release 1.4.03 (November 2001)
Quick SELECT internal buffer for output data enlarged to 64K to allow caching of large amount of data retrieved by single SQL SELECT.

5.4. Version 1.6

Release 1.6.02 (December 2002)

- Support Pro*C of Oracle 9.2
- Enhance 64 bit architecture support for Sun and HP machines.
- License mechanism added (See chapter on Quick SELECT Licensing)
- Introducing qselinfo utility (in the bin directory) to print Quick SELECT version information
- Product is packaged and delivered in two files: A PRODUCTION tar file which includes full functionality of the product but requires a license and a DEVELOPMENT tar file which does not require a license but no caching is performed. (See Administrator Guide Chapter)
- New sample make files were added to demo directory to allow application build with Pro*C 9 and Quick SELECT
- It is recommended to re-compile user applications after installing this release.
- Support AIX machines.

Release 1.6.03 (February 2004)

- Add the SUPPRESS_LOG variable that allows suppressing of license messages.

Release 1.6.04 (March 2005)

- Changing the licensing mechanism to have a license created for a version work for all its revisions (E.G: licenses created for 1.6.04 will work for 1.6.05, but not for 1.7.01).
- Adding support for long source files (over 8,192 lines in .pc file).
- Adding support for long SQL statements (over 8,192 characters long).
- Adding support for accessing a non-default database.

Release 1.6.05 (October 2005)
Fixing the mechanism to obtain IP address for license verification. The old mechanism used fork(), which caused defunc.

5.5. Version 1.7

Release 1.7.00 (July 2007)

- Features & Enhancements
  - QSEL was ported to Linux. At this phase, it was built and tested on Fedora Core 4 with Oracle client 9.2.0.4 & 10.2.0.1 and Oracle Server 9.2.0.4 (on i386 platform)

  - Some QSEL limits were removed:
    - Remove QSEL limit of 65K entries in host array size. Host arrays size are now limited to whatever Oracle Proc*C/server & compiler allow. QSEL does not validate the array size.

- Remove QSEL limit of 64K bytes for cached data size. Fetched data can be at any size and QSEL internal buffer will be dynamically extended when needed. The cached data size is now limited only by available virtual memory for the process and by maximum cache size defined by user.
- Remove QSEL limit of 1K bytes for total length of input host variables' values in a single statement. QSEL internal buffer for input host variables' values will be dynamically extended when needed. The input host variable' size is now limited only by available virtual memory for the process and by maximum cache size defined by user.
- Remove QSEL limit of 65K SELECT statements per process. The QSEL internal 16-bits statement ID field was extended to 32-bits allowing greater number of SQL statements per process.

- Enhancing the scope of table names candidate for caching: QSEL is now able to parse a SELECT statement text for table names not only in the first FROM clause but also in all other FROM clauses (in subqueries, unions etc). A new config file parameter "SUBQ" is introduced (or environment variable QSELSUBQ) which controls how QSEL will perform the table name parsing. Specifying a "N" value indicates that a simple parsing (only first FROM clause) is required. Specifying a "Y" value indicates that an extended parsing (all FROM clauses) is required. During an extended parsing, QSEL will accept a SELECT statement only when all table names in all FROM clauses are defined in appropriate TBNM config parameters. By default, QSEL will perform a simple parsing.

- QSEL Statistics:
  - QSEL statistics report was enhanced and contains new statistics about SQL activity and cache activity. The new report exposes some summary information that may help the application programmer/operator fine tune QSEL configuration in order to achieve better performance.
  - Auto statistics report: Starting at this release, there is no need to modify application source code in order to produce QSEL report. User can configure QSEL to automatically produce the statistics report at process exit. A new config file parameter "AUST" is introduced (or "QSELAUST" environment variable) that specifies whether a report should be produced before process terminates. Specify a "Y" value to create the report automatically. By default, QSEL will not produce the report automatically.

- QSEL Logging: A new, enhanced, verbose mode was added to allow debug information to be written to the application log file. This mode should be used when there is a need to reveal more information on QSEL behavior and/or when a problem requires a further investigation. The debug information will be produced only with the presence of a new QSEL environment variable "QSELDBG" which can have one or more of the
  following comma separated values:
  - "funccall" : will print a message on each function enter and exit.
  - "flow" : will print messages on major QSEL activities.
  - "license" : will print messages on license validation process.
  - "parse" : will print messages on the parsing phase of the SELECT statement.
  - "hostvar" : will print messages on the attributes and handling of host variables.
  - "cache" : will print messages on cache activities.
  - "dump" : will print messages related to various Pro*C structures that are analyzed by QSEL.
  - "all" : will print all above kind of messages..

  By default, no debug information will be produced.

- Fixed Bugs
  - Fixed bug in handling the combination of NULL columns/NULL constants with VARCHAR host variables. This bug caused QSEL not to handle the statement and access Oracle instead.
  - Fixed bug in handling of large sql statements. This bug caused QSEL not to handle the statement and access Oracle instead.

- QSEL Packaging: The product will be packaged for each combination of: platform , OS version, architecture. The Oracle version is no longer part of the packaging process of QSEL. A QSEL version packed for a certain combination of the above will run for any supported Oracle versions by QSEL resulting in less QSEL packages required to cover all needs. Internally, QSEL does not link to Oracle Client shared-

library anymore but loads it dynamically at runtime. It is assumed that there is a symbolic link to Oracle Client shared-library by the name of libclntsh.[so|sl] that points to the appropriate shared-library. In case there is no symbolic link with that name, the user MUST set the new QSEL environment variable "QSELORLB" with full path name of the Oracle Client shared-library.

- QSEL Documentation: QSEL User Guide includes a list of all run-time messages.

User Action Items
In order to use this QSEL release, user application source code does NOT need to be re-compiled but a new license should be requested and installed on each host running this QSEL version.

5.6. Version 1.8

- Added support for Oracle 11.1
- Fixed bug in referencing HP-UX Itanium shared-libraries with "so" suffix (during dynamic load).
- Fixed bug caused to core dump when host name was larger than 8 characters and /etc/hosts file did not contain an entry with host name.
- Fixed bug caused to error messages to come up when license was invalid and SQL statement was long and QSEL_SUPPRESS_LOG environment variable was set to "Y"

5.7. Version 1.9

- Added support for Oracle 11.2

Version 1.9.02

- Minor syntax change in qselmc.h header to support stricter compilers
- Fixed bug where setting QSELAUST variable to Y did not produce statistics. The bug had occurred only on Solaris

5.8. Version 2.0

- Added support for Red Hat Enterprise Linux Server release 6.4 platform (Santiago)

Version 2.0.00

- Corrected reference to web-site in the licensing program

5.9. Version 2.1 (**New**)

- Added support for Oracle 12.1. Release initially on Red Hat Enterprise Linux Server release 5.9 platform.

Version 2.1.00

6. Appendix: Performance Improvement Rate

6.1. Definition

The rate of improvement is the rate of NET saving of the CPU time consumed, for executing the SQL statements only, by both the process that issues the SQL statements and by the ORACLE server. Thus, the CPU time consumed by the rest of the program has to be disregarded, and the CPU time consumed by the ORACLE server has to be added.

The proper way to evaluate the rate of performance improvement is by measuring the CPU time of both the process issuing the SQL statements and the ORACLE server in 3 configurations:

- with normal ORACLE invocations (without QSEL)
- with QSEL
- with the ORACLE routine NULLIFIED (i.e., the "sqlcex" routine replaced with an EMPTY routine)

Let's abbreviate the above 3 configurations as "ORA", "QSEL" and "nul", respectively, and define the following:

nul_proc_CPU = the CPU time of "nul" process
nul_serv_CPU = the CPU time required by the server for "nul"
ORA_proc_CPU = the CPU time of "ORA" process
ORA_serv_CPU = the CPU time required by the server for "ORA"
QSEL_proc_CPU = the CPU time of "QSEL" process
QSEL_serv_CPU = the CPU time required by the server "QSEL"

nul_CPU = nul_proc_CPU + nul_serv_CPU
ORA_CPU = ORA_proc_CPU + ORA_serv_CPU
QSEL_CPU = QSEL_proc_CPU + QSEL_serv_CPU

QSEL_RATE_CPU = the rate of improvement in CPU

$$\texttt{QSEL\_RATE\_CPU} = 100 * \left(1 - \frac{\texttt{QSEL\_CPU - nul\_CPU}}{\texttt{ORA\_CPU - nul\_CPU}}\right)$$

Thus, finally,

Note, however, that such measurements cannot normally be performed with production programs, as these normally depend on the results of ORACLE accesses. Thus, the ORACLE routine, sqlcex, cannot be nullified as required.

6.2. Performance Improvement Rate

Within the QSEL target scope (see above), the rate of performance improvement is dependent mainly on the rate of repetitions. The higher the repetitions rate, the greater the performance improvement.

If the number of table rows referred to is significantly less (say, 2/3) than the minimum of (a) the number of rows in the table and (b) half the number of successful accesses, then a performance improvement rate of 50% can be expected. In the normal case, when a great number of successful accesses are made to relatively small tables, the rate can be even higher. The rate increases as, for a given table and number of accesses, as the number of table rows referred to decreases (i.e., a higher rate of repetitions).

For example, a performance improvement of at least 50% can be expected when, say, 10,000 accesses are made to a table even as large as 2,500 entries, when at least 85% of the accesses refer to at most 20% of the table rows. More accesses or a smaller table will result in a greater improvement, possibly reaching 80% or more.

6.3. SQL Not Within Scope

Some small processing time may be required for identifying, for a given SQL statement, whether it is within scope or not. Once a statement is flagged as not within scope, or is deactivated due to a problem, the overhead for it is very low. Note that even this small overhead can be eliminated by not compiling/linking with QSEL.

7. Appendix: Adjusting the Precompiler Makefiles

Normally, the makefile used for compiling/linking Pro*C programs is either the original makefile provided by ORACLE, or a derivative of it. A different makefile is provided by ORACLE for each version of Pro*C.

Any makefiles in use have to be duplicated and the new copies adjusted as described below, depending on the version of Pro*C in use.

7.1. Adjusting the Pro*C makefile

1. Add the following lines at the beginning of the makefile. Modify the QSEL_HOME according to your QSEL home directory.

   QSEL_HOME=/users/qsel
   QSELINCS=$(QSEL_HOME)/lib
   QSELSLIB=$(QSEL_HOME)/lib/qsel.sl
   QSELMODE='-I$(QSELINCS) $(QSELSLIB)'
   QSELPPOC=$(QSEL_HOME)/bin/qselppoc

2. Add ".qc" to the SUFFIXES list *between* ".c" and ".pc".

   .SUFFIXES: .exe .o .c **.qc** .pc

3. Duplicate the ".pc.c" suffix rule. In the *first* duplicate, replace the suffix rule by "pc.qc" and the suffix command by "$(QSELPPOC) < $*.pc > $*.qc". In the *second* duplicate, replace each occurrence of ".pc" with ".qc". Thus, the new ".pc.qc" suffix rule appears *before* the ".qc.c" suffix rule. For example (note that each indented line starts with a tab, not spaces):

   .pc.qc:
   	$(QSELPPOC) < $*.pc > $*.qc
   .**qc**.c:
   	$(PROC) $(PROFLAGS) iname=$*.**qc**

4. Duplicate the ".pc" suffix rule, and then:
   - In the *first* duplicate, insert at the first line "$(QSELPPOC) < $*.pc > $*.qc" and replace "iname=$*.pc" with "iname=$*.qc" on the second line.
   - In the *second* duplicate, replace each occurrence of ".pc" with ".qc". Thus, the new ".qc" suffix role appears *after* the ".pc" suffix rule.
   - Insert "$(QSELMODE)" at the ".pc" suffix rule command line, just before "$(PROLDLIBS)".
   - Insert "$(QSELMODE)" at the ".qc" suffix rule command line, just before "$(PROLDLIBS)".

   For example (note that each indented line starts with a tab, not spaces):

   .pc:
   	$(QSELPPOC) < $*.pc > $*.qc
   	-$(PROC) iname=$*.**qc** $(PROFLAGS)
   	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c **$(QSELMODE)** $(PROLDLIBS)
   .**qc**:
   	-$(PROC) iname=$*.**qc** $(PROFLAGS)
   	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c **$(QSELMODE)** $(PROLDLIBS)

6. Insert "$(QSELMODE)" at the ".c" suffix rule command line, just before "$(PROLDLIBS)". For example (note that each indented line starts with a tab, not spaces):

.c:
    @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c **$(QSELMODE)** $(PROLDLIBS)

7. **For Pro*C version 1.6 only:** Duplicate the ".pc.o" suffix rule. In the *first* duplicate, replace the suffix rule by "pc.qc" and the suffix command by "$(QSELPPOC) < $*.pc > $*.qc". In the *second* duplicate, replace each occurrence of ".pc" with ".qc". Thus, the new ".pc.qc" suffix rule appears *before* the ".qc.o" suffix rule. For example (note that each indented line starts with a tab, not spaces):

.pc.qc:
    $(QSELPPOC) < $*.pc > $*.qc
.**qc**.o:
    $(PROC16) $(PCCFLAGS) iname=$*.**qc**
    @$(ECHO) $(CC) $(CFLAGS) -c $*.c

Following is a listing of the original HP-UX Pro*C version 1.5 makefile for compilation/linkage of a Pro*C source program, with the modifications for QSEL embedded in **bold** typeface, and additions in ***bold italics***. The Pro*C version 1.6 and the SunOS makefiles are very similar, and at one point a version 1.6-only section is inserted for showing the adjustments.

```
#
# $Header: proc.mk.pp 7.26 93/07/26 13:50:28 seqdev Osd<unix> $ proc.mk.pp
#

#
# proc.mk - Command file for "make" to compile and load OCI and Pro*C programs.
#
# Pro*C programs are assumed to have the extension ".pc"
#
#
# Usage for sample OCI program:
#               make -f proc.mk sample
# Usage for sample Pro*C program:
#               make -f proc.mk samplec
#
```

Add the following lines here:

QSEL_HOME=/users/qsel
QSELINCS=$(QSEL_HOME)/lib
QSELSLIB=$(QSEL_HOME)/lib/qsel.sl
QSELMODE='-I$(QSELINCS) $(QSELSLIB)'
QSELPPOC=$(QSEL_HOME)/bin/qselppoc

```
SHELL=/bin/sh

LIBHOME=$(ORACLE_HOME)/lib

CC=cc

CFLAGS=-I. -O

CFLAGS=-I. -O -Aa -D_HPUX_SOURCE +ESsfc +ESlit

LDFLAGS=-Wl,-aarchive -L$(LIBHOME)

ARLOCAL=

AR=ar $(ARLOCAL)
ARCREATE=ar cr$(ARLOCAL)
ARDELETE=ar d$(ARLOCAL)
ARREPLACE=ar r$(ARLOCAL)

ECHO=$(ORACLE_HOME)/bin/echodo

PCC=$(ORACLE_HOME)/bin/pcc
PCCINC=$(ORACLE_HOME)/proc/lib
PCCFLAGS=include=$(PCCINC) ireclen=132 oreclen=132 select_error=no

PROC=$(ORACLE_HOME)/bin/proc
PROFOR=$(ORACLE_HOME)/bin/profor
PROCOB=$(ORACLE_HOME)/bin/procob
PROPAS=$(ORACLE_HOME)/bin/propas
PROADA=$(ORACLE_HOME)/bin/proada

PROFLAGS=ireclen=132 oreclen=132 select_error=no $(SQLCHECK) $(PROUSER)

LIBCORE=$(LIBHOME)/libcore.a

LIBCV6=$(LIBHOME)/libcv6.a

LIBNLSRTL=$(LIBHOME)/libnlsrtl.a

LLIBCORE=-lcore

LLIBCV6=-lcv6

LLIBNLSRTL=-lnlsrtl

LIBORA=$(LIBHOME)/libora.a
LLIBORA=-lora

LIBSQLDBA=$(LIBHOME)/libsqldba.a
```

```
LIBSQL=$(LIBHOME)/libsql.a

LIBOCIC=$(LIBHOME)/libocic.a

LIBPCC=$(LIBHOME)/libpcc.a

LIBPSD=$(LIBHOME)/libpsd.a

LIBOSDGEN=$(LIBHOME)/libosdgen.a

LIBPLUS=$(LIBHOME)/libsqlplus.a

LIBPLS=$(LIBHOME)/libpls.a

LIBKNL=$(LIBHOME)/libknl.a

LIBFORMS=$(LIBHOME)/libforms.a

LIBFORMS30=$(LIBHOME)/libforms30.a

LIBFORMS30P=$(LIBHOME)/libforms30p.a

LIBFORMS30C=$(LIBHOME)/libforms30c.a

LIBFORMS30X=$(LIBHOME)/libforms30x.a

LIBFORMS30M=$(LIBHOME)/libforms30m.a

LIBOKT=$(LIBHOME)/libokt.a

LIBOKTC=$(LIBHOME)/liboktc.a

LIBOKTX=$(LIBHOME)/liboktx.a

LIBOKTM=$(LIBHOME)/liboktm.a

OTHERLIBS=`cat $(ORACLE_HOME)/rdbms/lib/sysliblist` $(MLSLIBS)

LLIBPSO=

LLIBPSO=-lstublm

LIBSQLNET=$(LIBHOME)/libsqlnet.a
LLIBSQLNET=-lsqlnet

LIBTCP=$(LIBHOME)/libtcp.a

LIBNETWORK=$(LIBHOME)/libnetwork.a

LIBNETV2=$(LIBHOME)/libnetv2.a

LIBNTTCP=$(LIBHOME)/libnttcp.a

CONFIG=$(LIBHOME)/config.o

OSNTAB=$(LIBHOME)/osntab.o

OSNTABST=$(LIBHOME)/osntabst.o

FORMS30GUILIBS=

OSNTLPTB=$(LIBHOME)/osntlptb.o

NTCONTAB=$(LIBHOME)/ntcontab.o

CORELIBS=$(LLIBNLSRTL) $(LLIBCV6) $(LLIBCORE) $(LLIBNLSRTL) $(LLIBCV6) \
   $(LLIBCORE)
CORELIBD=$(LIBNLSRTL)  $(LIBCV6)  $(LIBCORE)

NETV2LIBS=$(LLIBSQLNET)
NETV2LIBD=$(LIBSQLNET)

NETLIBS=$(OSNTAB) $(LLIBSQLNET)
```

```
NETLIBD=$(OSNTAB) $(LIBSQLNET)

TTLIBD= $(NETLIBD)  $(LIBORA) $(CORELIBD)
TTLIBS= $(NETLIBS) $(LLIBORA) $(LIBPLSHACK) $(LLIBSQLNET) $(CORELIBS)\
   $(FORMS30GUILIBS) $(CLIBS)

PLSPECFILES=

STLIBS=$(OSNTABST) $(CONFIG) $(PLSPECFILES) -lora -lknlopt $(LIBPLS) -lknl \
      -lora -lknlopt $(LIBPLS) -lknl $(NETV2LIBS) -lora $(CORELIBS) \
      $(LLIBPSO) $(CLIBS)

STLIBD=$(LIBORA) $(LIBKNLOPT) $(LIBPLS) $(LIBKNL) $(NETV2LIBD) $(CORELIBD)

LIBTLI=$(OSNTLPTB) $(LLIBSQLNET)

PROLIBS=-locic -lsql

PCCLIBS=-lpcc -lsql -locic -lpcc

FRMLIBS=$(LIBHOME)/scp.o -lforms

TK2HOME = $(ORACLE_HOME)/guicommon/tk2

CTK2LIBS= $(LIBTK2C) $(LIBTK2OT) $(LIBTK2REM) $(LIBTK2RM) $(LIBTK2UT) \
      $(LIBTK2C) $(LIBTK2P) $(LIBTK2ROS) $(LIBTK2SL)

LIBCA = $(LIBHOME)/libca.a
LIBDE = $(LIBHOME)/libde.a
LIBGRAPHICS = $(LIBHOME)/libgraphics.a
LIBHH = $(LIBHOME)/libhh.a
LIBMMMM = $(LIBHOME)/mmmm.a
LIBMMOI = $(LIBHOME)/liboi.a
LIBMMOS = $(LIBHOME)/libos.a
LIBNN = $(LIBHOME)/libnn.a
LIBNNP = $(LIBNN) $(LIBHOME)/nntppb.o $(LIBHOME)/nntpps.o
LIBOACORE = $(LIBHOME)/liboacore.a
LIBOCL = $(LIBHOME)/libocl.a
LIBPLS11 = $(LIBHOME)/libpls11.a
LIBPSD11 = $(LIBHOME)/libpsd11.a
LIBRWS = $(LIBHOME)/librws.a
LIBSRWBM = $(LIBHOME)/libsrwbm.a
LIBSRWCM = $(LIBHOME)/libsrwcm.a
LIBSRWII = $(LIBHOME)/libsrwii.a
LIBSRWMV = $(LIBHOME)/libsrwmv.a
LIBSRWTO = $(LIBHOME)/libsrwto.a
LIBSSL = $(LIBHOME)/libssl.a
LIBTK2C = $(LIBHOME)/libtk2c.a
LIBTK2M = $(LIBHOME)/libtk2m.a $(LIBHOME)/libtk2p.a $(LIBHOME)/libtk2m.a
LIBTK2OT= $(LIBHOME)/libot.a
LIBTK2P = $(LIBHOME)/libtk2p.a
LIBTK2RE = $(LIBHOME)/libre.a
LIBTK2REM= $(LIBHOME)/librem.a
LIBTK2ROS = $(LIBHOME)/libros.a
LIBTK2SL = $(LIBHOME)/libsl.a
LIBTK2UC = $(LIBHOME)/libuc.a
LIBTK2UT = $(LIBHOME)/libut.a
LIBTK2UTIL = $(LIBHOME)/libutil.a
LIBTK2X = $(LIBHOME)/libtk2x.a
LIBVGS = $(LIBHOME)/libvgs.a
LIBZOM = $(LIBHOME)/libzom.a

XLIBS=-lXt -lX11 -lm

XLIBHOME = /usr/lib
MOTIFLIBHOME = /usr/lib
MOTIFLIBS = -L$(MOTIFLIBHOME) -lXm -L$(XLIBHOME) -lXt -lX11 -lm

CLIBS=$(OTHERLIBS)

#
# NOTE:   ORACLE_HOME must be either:
#                       . set in the user's environment
```

```
#                       . passed in on the command line
#                       . defined in a modified version of this makefile
#
```

Change the following line from:

```
.SUFFIXES: .exe .o .c .pc
```

To:

```
.SUFFIXES: .exe .o .c .qc .pc

USERID = scott/tiger

CFLAGS=-I. -O -Aa

#
# Define V6 and V7 specific macros here.
#
# Following macros contain ORACLE libraries that are linked to create:
# PROLDLIBS           : Pro*C application executables
# OCILDLIBS           : OCI application executables
# STPROLDLIBS         : single-task Pro*C application executables (no PL/SQL)
# STOCILDLIBS         : single-task OCI application executables (no PL/SQL)
# STPROLDLIBS_PLS : single-task Pro*C application executables (using PL/SQL)
# STOCILDLIBS_PLS : single-task OCI application executables (using PL/SQL)
#

LIBORAST=-lora -lknlopt -lpls -lpsd -lknl -lora -lknlopt -lpls -lpsd -lknl \
      $(NETV2LIBS) -lora
LIBORASTD=$(LIBORA) $(LIBKNLOPT) $(LIBPLS) $(LIBKNL) $(NETV2LIBD) $(LIBCORE)

LIBPROC= $(ORACLE_HOME)/proc/lib/libproc.a
LIBCGEN= $(ORACLE_HOME)/proc/lib/libcgen.a
PROCLIBS= $(LIBSQL) $(LIBPROC) $(LIBPCC) $(LIBCGEN) $(LIBOSDGEN) $(LIBPSD)
LIBPLSHACK= $(LIBPLS)

PROLDLIBS= $(LIBSQL) $(TTLIBS)
OCILDLIBS= $(LIBOCIC) $(TTLIBS)

STLDLIBS= $(STLIBS) $(LIBORAST) $(LIBCORE)
STPROLDLIBS= $(LIBSQL) $(STLDLIBS)
STOCILDLIBS= $(LIBOCIC) $(STLDLIBS)

STLDLIBS_PLS= $(STLDLIBS)
STPROLDLIBS_PLS= $(LIBSQL) $(STLDLIBS_PLS)
STOCILDLIBS_PLS= $(LIBOCIC) $(STLDLIBS_PLS)

all: sample samplec

sample: sample.o
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $@.o $(OCILDLIBS)

samplec: samplec.c samplec.pc
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $@ samplec.c $(PROLDLIBS)

samplest: sample.o
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $? $(STOCILDLIBS_PLS) $(CLIBS)

samplecst: samplec.c samplec.pc
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) samplec.c -o $@ $(STPROLDLIBS_PLS) $(CLIBS)
```

Add the following lines here:

```
.pc.qc:
        $(QSELPPOC) < $*.pc > $*.qc
```

Change the following lines from:

```
.pc.c:
        $(PROC) $(PROFLAGS) iname=$*.pc
```

To:

```
.qc.c:
        $(PROC) $(PROFLAGS) iname=$*.qc
```

```
LIBDIR= $(ORACLE_HOME)/proc/lib
DEMODIR= $(ORACLE_HOME)/proc/demo

install_files:
        -rm -f $(LIBDIR)/ORACA.H
        -rm -f $(LIBDIR)/SQLCA.H
        -rm -f $(LIBDIR)/SQLDA.H
        -rm -f $(DEMODIR)/proc.mk
        -ln $(LIBDIR)/oraca.h $(LIBDIR)/ORACA.H
        -ln $(LIBDIR)/sqlca.h $(LIBDIR)/SQLCA.H
        -ln $(LIBDIR)/sqlda.h $(LIBDIR)/SQLDA.H
        -ln $(LIBDIR)/proc.mk $(DEMODIR)/proc.mk

install: clean proc
        -chmod 755 $(ORACLE_HOME)/bin/proc
        -mv proc $(ORACLE_HOME)/bin/proc
        -chmod 755 $(ORACLE_HOME)/bin/proc

clean:
        -rm -f proc

proc: $(PROCLIBS) $(LIBORA) $(NETLIBD) $(LIBPLS) $(LIBPSD)
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $@  $(PROCLIBS) \
        $(LIBPLS) $(LIBPSD) $(TTLIBS)

procst: $(PROCLIBS) $(LIBORA) $(STLIBD) $(LIBPLS)
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $@  $(PROCLIBS) $(OSNTABST) \
        $(CONFIG) $(LIBORAST) $(LIBCORE) $(CLIBS)

singletask: procst
        -mv procst $(ORACLE_HOME)/bin/procst
        -chmod 755 $(ORACLE_HOME)/bin/procst

#
# General suffix rule to build executables from .pc and .c files.
#
# Usage :
#       make -f proc.mk USERID=<user/pass> <prog>
#
# For example to build an executable from a Pro*C source file named 'abc.pc'
# using scott/tiger for the ORACLE account name.  The make command line will
# be:
#       make -f proc.mk USERID=scott/tiger abc
#
# Note:  scott/tiger is the default account/password, so that you could
#       also use the following command line:
#
#       make -f proc.mk abc
#
# The executable will be named 'abc'.
#
```

Change the following lines from:

```
.pc:
        -$(PROC) iname=$*.pc $(PROFLAGS)
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c $(PROLDLIBS)
```

To:

```
.pc:
        $(QSELPPOC) < $*.pc > $*.qc
        -$(PROC) iname=$*.qc $(PROFLAGS)
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c $(QSELMODE) $(PROLDLIBS)
```

Add the following lines here:

```
.qc:
        -$(PROC) iname=$*.qc $(PROFLAGS)
        @$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c $(QSELMODE) $(PROLDLIBS)
```

Change the following lines from:

```
.c:
```

```
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c $(PROLDLIBS)
```

To:

```
.c:
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o $* $*.c $(QSELMODE) $(PROLDLIBS)
```

------ The following section applies to Pro*C 1.6 only: -------

Add the following lines here:

```
.pc.qc:
	$(QSELPPOC) < $*.pc > $*.qc
```

Change the following lines from:

```
.pc.o:
	$(PROC16) $(PCCFLAGS) iname=$*.pc
	@$(ECHO) $(CC) $(CFLAGS) -c $*.c
```

To:

```
.qc.o:
	$(PROC16) $(PCCFLAGS) iname=$*.qc
	@$(ECHO) $(CC) $(CFLAGS) -c $*.c
```

------ END of Pro*C 1.6 section ----------

```
#
# A Pro*C demo that that uses dynamic SQL to execute arbitray
# interactive SQL commands.
#

dsql: dsql.c dsql.pc
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) dsql.c -o $@ $(PROLDLIBS)

#
# Sample Pro*C demo programs with PL/SQL blocks.
#
PLSPCCFLAGS = ireclen=132 oreclen=132 sqlcheck=semantics userid=plsqa/supersecret

examp6: examp6.pc
	$(PROC) iname=$@.pc $(PLSPCCFLAGS)
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) examp6.c -o $@ $(PROLDLIBS)

examp7: examp7.pc
	$(PROC) iname=$@.pc $(PLSPCCFLAGS)
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) examp7.c -o $@ $(PROLDLIBS)

bankdemo:bankdemo.pc
	$(PROC) iname=$@.pc $(PLSPCCFLAGS)
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) bankdemo.c -o $@ $(PROLDLIBS)

sample5: sample5.pc
	$(PROC) iname=$@.pc $(PLSPCCFLAGS)
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) sample5.c -o $@ $(PROLDLIBS)

#
# Sample Pro*C user-exits.
#

UXTLDLIBS= $(ORACLE_HOME)/forms30/lib/iaddrvc.o \
	$(ORACLE_HOME)/forms30/lib/ifmdmf.o \
	$(ORACLE_HOME)/forms30/lib/ifplut.o \
	$(LIBFORMS30C) $(LIBFORMS30) $(LIBFORMS30P) $(LIBOKTC) $(LIBOKT) \
	$(ORACLE_HOME)/forms30/lib/libpls.a $(LIBFORMS30C) $(LIBFORMS30) \
	$(LIBOCIC) $(LIBSQL) $(TTLIBS)

concat: concat.pc
	$(PROC) $(PROFLAGS) iname=concat.pc
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o concat concat.c $(UXTLDLIBS)

sample5uxt: sample5uxt.pc
	$(PROC) iname=sample5uxt.pc $(PROFLAGS)
	@$(ECHO) $(CC) $(CFLAGS) $(LDFLAGS) -o sample5uxt sample5uxt.c \
	$(UXTLDLIBS)
```

8. Appendix: QSEL Run-time Messages

Normally, the makefile used for compiling/linking Pro*C programs is either the original makefile provided by ORACLE, or a derivative of it. A different makefile is provided by ORACLE for each version of Pro*C.

Any makefiles in use have to be duplicated and the new copies adjusted as described below, depending on the version of Pro*C in use.

8.1. INFO Level Messages

8.1.1. Module qselcex

SQL statement in file <file> line <n> accepted, ID=n

Description: QSEL accepted and will handle the SQL statement specified. The ID is an internal QSEL identification for the statement and will be then used internally by QSEL.

User Action: None.

WARN Level Messages

8.1.2. Module lpia

malloc() returned a non-NULL binary zero (0x00) pointer. %d bytes of storage will not be used

Description: cache memory allocation function was not able to allocate %d bytes of virtual memory. QSEL will continue trying to allocate the required amount of memory.

User Action: This message implies your parent application suffer from free store shortage. Examine your application and/or host memory configuration.

8.1.3. Module qselcex

SQL statement in file <file> line <n> will not be handled – not a SELECT statement

Description: QSEL will not handle the SQL statement specified because it is not a SELECT statement. QSEL will

User Action: None.

SQL statement in file <file> line <n> will not be handled – table list in FROM clause is too long

Description: QSEL will not handle the SQL statement specified because one of the FROM clauses in the statement defined a too long table name list. QSEL will forward the statement to be handled by Oracle.

User Action: Try to minimize the length of the problematic FROM clause.

SQL statement in file <file> line <n> will not be handled – FROM table(s) not specified in the control file

Description: QSEL will not handle the SQL statement specified because table list in a FROM clauses does not match any of the table lists configured in the control file. QSEL will forward the statement to be handled by Oracle.

User Action: If SELECT statement fetches from update-insensitive table(s) – consider adding these to QSEL control file so that SELECT result will be cached.

SQL statement in file <file> line <n> will not be handled – FOR UPDATE clause specified

Description: QSEL will not handle the SQL statement specified because the SELECT statement contains the “FOR UPDATE” phrase, which inhibits caching. QSEL will forward the statement to be handled by Oracle.

User Action: None.

SQL statement in file <file> line <n> will not be handled – SYSDATE function used

Description: QSEL will not handle the SQL statement specified because the SELECT statement contains the “SYSDATE” function, which is inhibits caching. QSEL will forward the statement to be handled by Oracle.

User Action: None.

SQL statement in file <file> line <n> will not be handled – CURRVAL or NEXTVAL pseudo column used

Description: QSEL will not handle the SQL statement specified because the SELECT statement uses pseudo columns that are not cacheable. QSEL will forward the statement to be handled by Oracle.

User Action: None.

SQL statement in file <file> line <n> will not be handled – host variable (n) pointers to strings or to "extern char[]" without a length not supported

Description: QSEL will not handle the SQL statement specified because host variable n (n >= 0) is a string pointer or an extern char which is not supported by QSEL. QSEL will forward the statement to be handled by Oracle.

User Action: Refer to "Target SQL" section for more information on QSEL limits.

SQL statement in file <file> line <n> will not be handled – host variable (n) unsupported host variable type (t)

Description: QSEL will not handle the SQL statement specified because host variable n (n >= 0) is of type t which is not supported by QSEL. Usually, this message indicates that the corresponding column type is not supported. QSEL will forward the statement to be handled by Oracle.

User Action: Refer to "Target SQL" section for more information on QSEL limits.

SQL statement in file <file> line <n> will not be handled – caching is limited to PRODUCTION version only

Description: QSEL will not handle the SQL statement specified because the product (QSEL) was built in such a mannger that skips license validation phase. QSEL will forward the statement to be handled by Oracle. This message indicates that user installed wrong QSEL delivery file or that QSEL delivery file was not created properly.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – more than n SQL statements

Description: QSEL will not handle the SQL statement specified because the product (QSEL) is limited to handle up to n SQL SELECT statements per process. QSEL will forward the statement to be handled by Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – unexpected SQLSTATE value (val1)

Description: QSEL will not handle the SQL statement specified because Oracle returned an unexpected value val1 in SQLSTATE. For future calls, QSEL will forward the statement to be handled by Oracle.

User Action: Check and if possible fix the reason for this SQL statement failure.

SQL statement in file <file> line <n> will not be handled – unexpected SQLCODE value (val1) for MODE=ORACLE

Description: QSEL will not handle the SQL statement specified because Oracle returned an unexpected value val1 in SQLCODE. For future calls, QSEL will forward the statement to be handled by Oracle.

User Action: Check and if possible fix the reason for this SQL statement failure.

SQL statement in file <file> line <n> will not be handled – unexpected SQLCODE value (val1)

Description: QSEL will not handle the SQL statement specified because Oracle returned an unexpected value val1 in SQLCODE. For future calls, QSEL will forward the statement to be handled by Oracle.

User Action: Check and if possible fix the reason for this SQL statement failure.

SQL statement in file <file> line <n> will not be handled – SQL Warning condition. Sqlwarn=string

Description: QSEL will not handle the SQL statement specified because Oracle returned a warning. For future calls, QSEL will forward the statement to be handled by Oracle.

User Action: Check and if possible fix the reason for this SQL statement failure.

SQL statement in file <file> line <n> will not be handled – unexpected lpiasr return code (rc)

Description: QSEL will not handle the SQL statement specified because QSEL cache search routine returned an unexpected code rc. For future calls, QSEL will forward the statement to be handled by Oracle.

User Action: Call QSEL Support Team.

Quick-SELECT disabled via QSELDSAB

Description: During initialization, QSEL encountered that it was configured to be disabled (via environment variable QSELDSAB) and therefore will be disabled.

User Action: None

"QSELDSAB=N" specified. "DSAB" value in control file record number n. Record ignored

Description: During initialization, dual configuration was found via both QSELDSAB environment variable and DSAB control file parameter. QSEL took into considiration the environment variable configuration.

User Action: None

Quick-SELECT disabled via the control file

Description: During initialization, QSEL encountered that it was configured to be disabled (via control file parameter DSAB) and therefore will be disabled.

User Action: None

SQL statement in file <file> line <n> will not be handled – Not enough storage for segments of SELECT statement text

Description: During handling a SELECT statement text segments created by Oracle Precompiler, QSEL failed to allocate enough memory to concatenate all segments of SQL text so it can be parsed. QSEL will forward this SQL to Oracle.

User Action: Check the application memory usage and see if some memory can be freed.

Quick-SELECT license will expire in n days. Obtain new license.

Description: During initialization, QSEL encountered that its installed license will expire n days from now.

User Action: Obtain a new license as soon as possible.

Quick-SELECT license will expire in TODAY. Obtain new license.

Description: During initialization, QSEL encountered that its installed license will expire TODAY.

User Action: Obtain a new license as soon as possible.

Quick-SELECT license has already expired and grace period will end in n days. Obtain new license.

Description: QSEL license has already expired and now is in grace period time. However, this grace period will expire too in n days and after that QSEL will be disabled.

User Action: Obtain a new license as soon as possible.

8.2. ERR Level Messages

8.2.1. Module orastub

unable to load shared-library %s specified by environment variable QSELORLB

Description: QSEL was not able to load Oracle-Client shared-library as specified in the QSELORLB environment variable. QSEL will attempt to load the default library.

User Action: Check that appropriate shared-library and path is specified in the QSELORLB environment.

8.2.2. Module qselcex

SQL statement in file <file> line <n> will not be handled – Not enough storage for copying statement text

Description: QSEL will not handle the SQL statement specified because it could not allocate enough memory for copying the SELECT text string for parsing. QSEL will forward the statement to be handled by Oracle.

User Action: Check the amount of memory consumed by your application and try to free some unnecessary dynamic allocations.

SQL statement in file <file> line <n> will not be handled – unable to allocate internal key buffer at size of n bytes

Description: QSEL will not handle the SQL statement specified because it could not allocate enough memory for the key element of the cache entry. QSEL will forward the statement to be handled by Oracle.

User Action: Check the amount of memory consumed by your application and try to free some unnecessary dynamic allocations.

SQL statement in file <file> line <n> will not be handled – unable to allocate internal data buffer at size of n bytes

Description: QSEL will not handle the SQL statement specified because it could not allocate enough memory for the data element of the cache entry. QSEL will forward the statement to be handled by Oracle.

User Action: Check the amount of memory consumed by your application and try to free some unnecessary dynamic allocations.

SQL statement in file <file> line <n> will not be handled – actual length (len1) of host variable [hid][rid] exceeds its maximum length (len2)

Description: QSEL will not handle the SQL statement specified because the actual length len1 of host variable nmber hid (hid >= 0) in row number rid exceeds the maximum length len2 allowed for this variable. QSEL will not try to handle this statement anymore and on subsequent calls it will be immeidately forwarded to Oracle.

User Action: This message indicates that there might be a problem in Oracle or in QSEL. Try to run the program with QSEL disabled and see the results. If program runs smoothly without QSEL then there is a possible bug in QSEL.

SQL statement in file <file> line <n> will not be handled – Concatenation of (type1) host variable fields too long - exceeds the limit of (len1)

Description: QSEL will not handle the SQL statement specified because the actual length concatenating all host variables of type type1 exceeds the length len1 that was calculated when statement was parsed. QSEL will not try to handle this statement anymore and on subsequent calls it will be immeidately forwarded to Oracle.

User Action: If this program runs smoothly when QSEL is disabled then it is probably a QSEL bug.s

Invalid QSELSVLV - value ignored

Description: During initialization, QSEL encountered bad value specified in the QSELSVLV environment variable and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid QSEL_SUPPRESS_LOG- value invalid

Description: During initialization, QSEL encountered bad value specified in the QSEL_SUPPRESS_LOG environment variable and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid QSELMXSG - value ignored

Description: During initialization, QSEL encountered bad value specified in the QSELMXSG environment variable and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid QSELAVLN - value ignored

Description: During initialization, QSEL encountered bad value specified in the QSELAVLN environment variable and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid QSELSUBQ - value invalid. "N" assumed

Description: During initialization, QSEL encountered bad value specified in the QSELSUBQ environment variable and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid QSELAUST - value invalid. "N" assumed

Description: During initialization, QSEL encountered bad value specified in the QSELSUBQ environment variable and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Table name list from control file record number n truncated. Record ignored

Description: During initialization, QSEL encountered too long table name list in a TBNM control file parameter at the specified record. QSEL ignored this list.

User Action: Shrink the size of the long list.

Not enough storage for table name list from control file record number n. Record ignored

Description: During initialization, QSEL could not allocate enough memory to hold the table list that is specified in the control file at record n. As a result, QSEL ignored that table list.

User Action: Shrink the size of the long list or free some memory.

Invalid AVLN value in control file record number n. Record ignored

Description: During initialization, QSEL encountered bad value specified in the AVLN control file parameter and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid MXSG value in control file record number n. Record ignored

Description: During initialization, QSEL encountered bad value specified in the MXSG control file parameter and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid SVLV value in control file record number n. Record ignored

Description: During initialization, QSEL encountered bad value specified in the SVLV control file parameter and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid SUPRESS_LOG value in control file record number n. "N" assumed (Quick-SELECT license messages will be output)

Description: During initialization, QSEL encountered bad value specified in the SUPRESS_LOG control file parameter and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid SUBQ value in control file record number n. "N" assumed

Description: During initialization, QSEL encountered bad value specified in the SUBQ control file parameter and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Invalid AUST value in control file record number n. "N" assumed

Description: During initialization, QSEL encountered bad value specified in the AUST control file parameter and a default value was used instead.

User Action: Correct the QSEL configuration parameter and rerun.

Unrecognized keyword in control file record number n. Record ignored

Description: During initialization, QSEL encountered an unrecognized control file parameter at the specified record number and ignored it.

User Action: Correct the QSEL configuration parameter and rerun.

SQL statement in file <file> line <n> will not be handled – insert in cache failed (RC=n)

Description: QSEL will not handle the SQL statement specified because it could not insert the data into the cache. Cache component return code is n. QSEL will forward the statement to be handled by Oracle.

User Action: If RC=3003, a single statement requires more memory than the total cache memory allocated. Either increase cache size (MXSG), or do not include this table combination in the TBNM list in the configuration file. If RC<>3003, call QSEL Support Team.

8.3. SEVERE Level Messages

8.3.1. Module lpia

Amount of declared max storage (%d) not available. malloc(%) failed

Description: cache memory allocation function was not able to allocate %d bytes of virtual memory and therefore the maximum amount of storage defined for cache will not be allocated.

User Action: This message implies your parent application suffer from free store shortage. Examine your application and/or host memory configuration.

8.3.2. Module qselcex

SQL statement in file <file> line <n> will not be handled - invalid QSEL_Global version

Description: QSEL will not handle the SQL statement specified because of unexpected value in QSEL global structure that is passed on each invocation. QSEL will forward the statement to be handled by Oracle. This problem can occur when QSEL distribution files are wrong or when user application performed illegal operation which caused memory overrun.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled - invalid QSEL_Global language

Description: QSEL will not handle the SQL statement specified because QSEL global structure indicates that QSEL was called from program written in language other than Pro*C. QSEL will forward the statement to be handled by Oracle. This problem can occur also when QSEL distribution files are wrong or when user application performed illegal operation which caused memory overrun.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled - Program precompiled by a version of Pro*C/Pro*COBOL unsupported by QSEL

Description: QSEL will not handle the SQL statement specified because user program was precompiled with Oracle Precompiler version that is not supported by QSEL. QSEL will forward the statement to be handled by Oracle. This problem can occur also when QSEL distribution files are wrong or when user application performed illegal operation which caused memory overrun.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled – no valid license exists

Description: QSEL will not handle the SQL statement specified because there is no valid QSEL runtime license installed on the host. QSEL will forward the statement to be handled by Oracle.

User Action: Obtain and install valid license.

SQL statement in file <file> line <n> will not be handled – No valid key is installed in license file

Description: QSEL will not handle the SQL statement specified because the QSEL runtime license key is not valid for this QSEL version and/or this host. QSEL will forward the statement to be handled by Oracle.

User Action: Obtain and install valid license.

SQL statement in file <file> line <n> will not be handled – License could not be validated. <msg>

Description: QSEL will not handle the SQL statement specified because some problem occurred during validation of the license key file. <msg> will specify more information on the reason of this failure. QSEL will forward the statement to be handled by Oracle.

User Action: If <msg> is clearifies the problem, try to fix it. If not - Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – License file name could not be determined. <msg>

Description: QSEL will not handle the SQL statement specified because some problem occurred while trying to locate and open the license key file. <msg> will specify more information on the reason of this failure. QSEL will forward the statement to be handled by Oracle.

User Action: If <msg> is clearifies the problem, try to fix it. If not - Contact QSEL Support Team.

8.4. FATAL Level Messages

8.4.1. Module orastub

unable to load function %s from Oracle Client shared-library. (%s)

Description: QSEL was not able to load a symbol from Oracle-Client shared-library.

User Action: Check that appropriate shared-library is installed and there is a correct symbolic link to it. You may need to configure QSEL to search Oracle-Client shared library elsewhere. Refer to "Control Definitions Files" and "Environment Variables" sections.

unable to load shared-library %s

Description: QSEL was not able to load the default Oracle-Client shared-library. QSEL will not be able to interact with Oracle and user application will not run.

User Action: Check that Oracle-Client shared-library has a symbolic link named libclntsh.[so|sl]. If not, create one or use the QSELORLB environment variable to configure QSEL to the exact shared-library to use.

8.4.2. Module qselcex

"QSELDSAB" value invalid. "Y" assumed (Quick-SELECT disabled)

Description: During initialization, QSEL encountered that QSELDSAB environment variable has invalid value and therefore the default action was taken: to disable Quick-SELECT.

User Action: Verify that this is the behavior you want. If not, fix the value of the environment variable.

Control file <name> cannot be opened - Quick-SELECT disabled

Description: During initialization, QSEL could not open control file.

User Action: Verify that QSELCTDF environment variable points to the correct control file.

Invalid DSAB value in control file record number n. "Y" assumed (Quick-SELECT disabled)

Description: During initialization, QSEL encountered an invalid value in DSAB control file parameter and therefore the default action was taken: to disable Quick-SELECT.

User Action: Verify that this is the behavior you want. If not, fix the value of the environment variable.

No table names combination specified/accepted

Description: During initialization, QSEL encountered that no table names list were specified in the control file or that some table names lists were specified but ignored by QSEL and therefore QSEL will run without any table names list which will cause caching to be disabled.

User Action: Verify that this is the behavior you want. If not, specify/fix the value of TBNM parameter.

Init of cache memory failed (RC=n)

Description: During initialization, QSEL failed to initialize the cache mechanism component which returned an error code n.

User Action: Contact QSEL Support Team.

8.5. BUG Level Messages

8.5.1. Module lpia

lpiaal invoked before lpiait

Description: cache memory allocation function was called before cache initialization function.

User Action: This error implies that there is a bug in QSEL code. Contact QSEL Support team.

8.5.2. Module qselcex

SQL statement in file <file> line <n> will not be handled – sqlstm is NULL

Description: QSEL will not handle the SQL statement specified because it can not access Oracle Precompiler generated structure sqlstm. QSEL will forward the statement to be handled by Oracle. This problem can occur when QSEL distribution files are wrong or when user application performed illegal operation which caused memory overrun, or when there is a bug in Oracle Precompiler.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled – cannot analyze statement

Description: QSEL will not handle the SQL statement specified because it can not access the specific SQL statement information that is stored in Oracle Precompiler generated structures. QSEL will forward the statement to be handled by Oracle. This problem can occur when QSEL distribution files are wrong or when user application performed illegal operation, which caused memory overrun, or when there is a bug in Oracle Precompiler.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled – Opcode of SELECT, but sqlstm->stmt is NULL

Description: QSEL will not handle the SQL statement specified although it is a SELECT statement because it cannot access the SQL text stored in Oracle Precompiler generated structures. QSEL will forward the statement to be handled by Oracle. This problem can occur when QSEL distribution files are wrong or when user application performed illegal operation, which caused memory overrun, or when there is a bug in Oracle Precompiler or in QSEL.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled – Validation of Pro*C variables failed

Description: QSEL will not handle the SQL statement specified although it is a SELECT statement because it cannot analyze some of the fields stored in Oracle Precompiler generated structures. QSEL will forward the statement to be handled by Oracle. This problem can occur when QSEL distribution files are wrong or when user application performed illegal operation, which caused memory overrun, or when there is a bug in Oracle Precompiler or in QSEL.

User Action: Try to run the qselsamp demo program that is supplied with QSEL. If it fails with same message then you should contact QSEL Support Team. If it runs smoothly then it is likely that there is something wrong with user application.

SQL statement in file <file> line <n> will not be handled – Opcode of SELECT, but statement text does not start with "SELECT"

Description: QSEL will not handle the SQL statement specified due to a collision in Oracle Precompiler generated structures fileds: one one hand the statement is marked with an internal Oracle opcode as a SELECT statement but the SQL text does not begin with "SELECT" phrase. QSEL will forward the statement to be handled by Oracle. This problem can occur when there is a bug in Oracle Precompiler or in QSEL.

User Action: Contact QSEL Support Team in order to verify where the problem is.

SQL statement in file <file> line <n> will not be handled – FROM clause not found

Description: QSEL will not handle the SQL statement specified because QSEL parser could not find the FROM clause in the SELECT statement. QSEL will forward the statement to be handled by Oracle. This problem can occur when there is a bug in Oracle Precompiler or in QSEL.

User Action: Contact QSEL Support Team in order to verify where the problem is.

SQL statement in file <file> line <n> will not be handled – table list in FROM clause is empty

Description: QSEL will not handle the SQL statement specified because QSEL parser could not find any table name list in a FROM clause in the SELECT statement. QSEL will forward the statement to be handled by Oracle. This problem can occur when there is a bug in Oracle Precompiler or in QSEL.

User Action: Contact QSEL Support Team in order to verify where the problem is.

SQL statement in file <file> line <n> will not be handled – host variable (n) pointer is NULL

Description: QSEL will not handle the SQL statement specified because it could not locate the pointer to host variable n (n >=0 where n=0 indicated the 1st host variable). QSEL will forward the statement to be handled by Oracle. This problem can occur when there is a bug in QSEL.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – host variable (n) neither input nor output

Description: QSEL will not handle the SQL statement specified because host variable n (n >=0 where n=0 indicated the 1st host variable) is marked by Oracle Precompiler generated structures neither as input nor output type. QSEL will forward the statement to be handled by Oracle. This problem can occur when there is a bug in QSEL or in Oracle Precompiler.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – INTO clause missing or does not specify any host variable

Description: QSEL will not handle the SQL statement specified because there is no INTO clause in the statement or it contains no host variables. Although it might be a valid SELECT, QSEL requires an INTO clause. QSEL will forward the statement to be handled by Oracle.

User Action: Add valid INTO clause.

SQL statement in file <file> line <n> will not be handled – prefix len (len1) > max len (len2)

Description: QSEL will not handle the SQL statement specified because of internal QSEL bug specifying length field of some prefix field that is preceeding each cache key field. QSEL will forward the statement to be handled by Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – number of rows processed by oracle (n) exceeds number of rows requested by program (m)

Description: QSEL will not handle the SQL statement specified because sqlca.sqlerrd[2] specifies that oracle processed much more rows then specified in statement. QSEL will forward the statement to be handled by Oracle.

User Action: Contact Oracle Support.

SQL statement in file <file> line <n> will not be handled – Missing SQLCA

Description: QSEL will not handle the SQL statement specified because it cannot locate the SQLCA structure. QSEL will forward the statement to be handled by Oracle.

User Action: Add the SQLCA definition to the program and re-build it.

SQL statement in file <file> line <n> will not be handled – NULL sqlstm or sqlstm->cud or sqlstm->sqlest

Description: QSEL will not handle the SQL statement specified because it cannot access some important fields in Oracle Precompiler generated headers. QSEL will forward the statement to be handled by Oracle.

User Action: Contact QSEL Support.

SQL statement in file <file> line <n> will not be handled – unexpected sqlety value (val1), QSEL does not know how to read Oracle return code

Description: QSEL will not handle the SQL statement specified because it cannot determine how to read Oracle return code. QSEL will not handle this statement in future calls.

User Action: This message indicates some new Oracle functionality not supported by QSEL or a QSEL bug. Contact QSEL Support.

SQL statement in file <file> line <n> will not be handled – SQLCODE address not set

OR

SQL statement in file <file> line <n> will not be handled – Neither SQLCODE nor SQLSTATE addresses set

Description: QSEL will not handle the SQL statement specified because it cannot access one or more of Oracle return code/state structures. QSEL will not handle this statement in future calls.

User Action: Verify that application program defines the appropriate structures. If problem still exists then it is probably a QSEL bug.

SQL statement in file <file> line <n> will not be handled – Opcode of SELECT, but statement text does not start with "SELECT" (detected by qselbuf)

Description: During handling a SELECT statement text segments created by Oracle Precompiler, QSEL encountered that although statement is marked as a SELECT statement in Oracle Precompiler generated structures, the text segments of SQL text do not begin with the word “SELECT”. QSEL will forward this SQL to Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – Invalid QSEL Buf_Mode (c) in qselcex

Description: During handling a SELECT statement text segments created by Oracle Precompiler, QSEL encountered an unexpected value in its internal segments concatenation mechanism. QSEL will forward this SQL to Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – Mismatch of SELECT statement segmented text lengths: Coded: len1, actual: len2, last chunk: len3

Description: During handling a SELECT statement text segments created by Oracle Precompiler, QSEL encountered a mismatch in the length of the entire SELECT text string. The mismatch is between the length (len1) that is coded in Oracle Precompiler generated structures and the actual length (len2) calculated by QSEL. The len3 field indicated the length of the last segment processed by QSEL. QSEL will forward this SQL to Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – during concat of SELECT statement text segments - sum of the segments does not match coded length

Description: Although correctly pre-calculated, the actual process of concating all segments of SELECT text yielded wrong length which does not match the length coded into Oracle Precompiler generated structures. QSEL will forward this SQL to Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – Cached number of rows exceeds sqlstm.iters

Description: QSEL will not handle the SQL statement specified because the number of rows stored in the cached element is greater than maximum defined by user application. QSEL will forward this SQL to Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – Cached length (len1) of host variable [hid][rid] exceeds its maximum length (len2)

Description: QSEL will not handle the SQL statement specified because the cached length len1 of host variable number hid (hid >= 0) in row number rid exceeds the maximum length len2 allowed for this variable. QSEL will not try to handle this statement anymore and on subsequent calls it will be immeidately forwarded to Oracle.

User Action: Contact QSEL Support Team.

SQL statement in file <file> line <n> will not be handled – Actual cache data length (len1) does not match the coded length (len2)

Description: QSEL will not handle the SQL statement specified because not all content of cached data was copied to application host variables. QSEL will not try to handle this statement anymore and on subsequent calls it will be immeidately forwarded to Oracle.

User Action: Contact QSEL Support Team.