NAME

`vertrace` – *Automatically* trace and display a profiled SQLcl experience

SYNOPSIS

`vertrace` *command* [*options*] [*parameters*]

DESCRIPTION

SQLcl is Oracle's modern feature-rich replacement for their SQL*Plus command.

The vertrace SQLcl extension lets you *easily* see a beautiful receipt for response time™ with pinpoint accuracy for your Oracle code.

Any vertrace command requiring access to the database will require the `XXJHVT_VERTRACE_DEVEL` role unless otherwise noted.

PROCESS

Your `vertrace` commands control when tracing is turned on and off. When tracing is turned off your tracefile is automatically fetched and profiled. Profiles are opened according to the current variable settings. The `client` and `module` commands can create multiple tracefiles and those too will be processed the same way when their standing orders are canceled.

COMMANDS

`client` **[-off]** **[-list]** **[*clientId*]** Give a standing order to trace any session's activity tagged with the given *clientId*. Then notify the monitoring thread to periodically query active traces that match that predicate.

Use `-off` *clientId* to cancel the prior order and process the tracefiles created by that order.

Use `-list` to list all client orders given by this session that haven't been canceled.

Sessions are matched for values in column `CLIENT_IDENTIFIER` in `V$SESSION`.

Requires the `XXJHVT_VERTRACE_ADMIN` role.

`fetch` ***remoteFilename*** Fetch an arbitrary tracefile from the database server and then process it.

Requires the `XXJHVT_VERTRACE_ADMIN` role.

`foreach` The `foreach` directive says to create one tracefile for every traceable statement that is executed until you execute `vertrace off`, resulting in multiple tracefiles, each having one traced statement and each being processed similarly.

`help` Open the help pdf using the operating system's default opener. Do not use the SQLcl builtin `help` command.

`history` Show a list of `vertrace` commands executed in this SQLcl session, in execution order.

`module` [**-off**] [**-list**] [**-action** ***action***] [***module***] Give a standing order to trace any session's activity tagged with the given *module* and optionally the given *action*. Then notify the monitoring thread to periodically query active traces that match that predicate.

Use `-action` *action* to narrow the trace to an *action* within a *module*.

Use `-off` [-action *action*] *module* to cancel the prior order and process tracefiles created by that order.

Use `-list` to list all module orders given by this session that haven't been canceled.

Sessions are matched for values seen in columns `MODULE` and `ACTION` in `V$SESSION`.

Requires the `XXJHVT_VERTRACE_ADMIN` role.

`next` Trace only the next traceable statement executed in SQLcl, resulting in one tracefile containing the traced statement. And then process it. Tracing is enabled immediately before its execution and disabled immediately afterward.

`off` Turns off trace to cancel the effect of any `on` or `foreach` command. All tracefiles identified will be processed.

`on` Turn on trace immediately for this session. Tracing remains on until you execute `vertrace off`. All traced activity will be in one tracefile.

`open` ***localFilename*** Process or reprocess the given tracefile in a local filesystem.

`save` Save variables that changed since SQLcl started.

`set` ***variable value*** Set *variable* to *value*.

If you want the change to persist, then execute the `save` command.

`sid` [**-off**] [**-list**] [***integer***] Enable trace for the specified session with SID = *integer*, for a row matching `V$SESSION.SID`.

Use `-off` *integer* to cancel the prior order trace and process the tracefile according to the current variable settings.

Use `-list` to list all sid orders given by this session that haven't been canceled.

Requires the `XXJHVT_VERTRACE_ADMIN` role.

`show` ***variable*** Show a variable named *variable*.

Use the special keyword `all` to see all variables. Variable names prefixed with "*" have a value different from what is persisted.

`stage` Copy the administration scripts to `~/.sqlcl/vertrace/admin` for reviewing, editing, and execution. Installation instructions are within `xxjhvt_vertrace_install.sql`.

`status` Display the current status of this extension.

`until` Obsolete. This synonym for the `on` command will be removed in a future release.

`version` Print the version of this extension.

VARIABLES

Variables are stored in `~/.sqlcl/vertrace/vertrace.properties`. Variable names are case-insensitive. Boolean variables take a value in (`on`,`off`) which is case-insensitve. Filename variables take a name that should exist in a local filesystem and case sensitivity depends on the filesystem. The `set` command will print a warning if the file does not exist.

`BINDS` **(boolean)** If `on`, then enable the collection of placeholder values in the trace data. The default value is `off`.

`DEBUG` **(boolean)** If `on`, then debugging information is written to the console revealing important context and progress. The default value is `off`.

`MRPROF` **(filename)** The absolute path to the mrprof executable that is part of the Workbench. There is no default value.

`MRPROF_OPTS` **(string)** These options are added to the mrprof command line after all other options are added. The default value is `--noelide --noforce-match-statement-texts`.

`MRPROF_WAIT` **(boolean)** If `on`, then wait for the tracefile to be fetched, profiled, and opened (if applicable) before allowing the user to continue. Otherwise, control returns immediately to the user after tracing is turned off. The default value is `on`.

`OPEN` **(boolean)** If `on`, then `--open` is added to the mrprof command line resulting in the profile for a local tracefile being opened by the default web browser. This is a convenience variable applied before `MRPROF_OPTS` is processed. The default value is `on`.

`PLANSTAT` **(string)** Determines how frequently row source execution statistics are collected in the trace data. Valid values are in (`all_executions`, `first_execution`, `never`). The default value is `first_execution`.

`SHOW_MIE` **(boolean)** If `on`, the commands that `vertrace` executes to enable and disable tracing are shown in the trace data. Otherwise, the pinpoint accuracy applies. The default value is `off`.

`STATS` **(string)** Set the Oracle session's `STATISTICS_LEVEL` parameter to this value before tracing and restore its prior value after tracing. Valid values are in (`all`, `basic`, `typical`). The default value is `typical`.

See the Oracle documentation for more information.

`TIMING` **(boolean)** If `on`, the elapsed time of the experience is printed. This differs from a value printed by the SQLcl `set timing on` command. The experience time does not include the extra commands `vertrace` executes before and after your commands. You can think of this time as what `set timing on` would have printed had `vertrace` not been employed. The default value is `off`.

`TRCDIR` **(filename)** The directory where tracefiles will be written. The default value is `~/.sqlcl/vertrace/traces`.

`WATCH_MILLI` **(integer)** When giving a standing order to trace a module or a client, this determines how long to wait between polling `V$SESSION` and `V$PROCESS` for evidence of its effect. The smaller the number, the more frequent the polling. The default value is 5000 milliseconds.

EXAMPLES

Some whitespace in the output is compressed for convenience here.

1. First ever execution fails showing an error while the second succeeds.

```
SQL> vertrace next
```

```
SQL> select * from dual;

DUMMY
--------
X

vertrace: wrote 4189 bytes to ~/.sqlcl/vertrace/traces/EBSCDB_ora_27026-1.trc
vertrace: error: cannot process tracefile: Set the MRPROF variable to a working
Method R Workbench Profiler command. Then VERTRACE OPEN the file that failed and
consider VERTRACE SAVE.
SQL> vertrace show MRPROF
vertrace: MRPROF  <null>
SQL> !find $HOME -name mrprof
/home/gpapado/wb/bin/mrprof
SQL> vertrace set mrprof "/home/gpapado/wb/bin/mrprof"
SQL> vertrace next
SQL> select * from dual;

DUMMY
--------
X

vertrace: wrote 3769 bytes to ~/.sqlcl/vertrace/traces/EBSCDB_ora_27026-2.trc
SQL> vertrace save
```

2. Trace two traceable statements, producing two tracefiles, disable tracing, and then execute a statement that's not traced.

```
SQL> vertrace foreach
SQL> select count(*) from all_objects;

   COUNT(*)
-----------
      56601

vertrace: wrote 3094058 bytes to ~/.sqlcl/vertrace/traces/EBSCDB_ora_8724-1.trc
SQL> desc dual

  Name     Null?    Type
  -------- -------- --------------
  DUMMY             VARCHAR2(1)
vertrace: wrote 351007 bytes to ~/.sqlcl/vertrace/traces/EBSCDB_ora_8724-2.trc
SQL> vertrace off
SQL> select * from dual;

DUMMY
--------
```

```
X
SQL>
```

3. Trace a script and all its nested calls

```
SQL> !cat x.sql
select 'before y' from dual;
@y
select 'after y' from dual;
SQL> !cat y.sql
select 'before z' from dual;
@z
select 'after z' from dual;
SQL> !cat z.sql
select 'in z' from dual;
SQL> vertrace next
SQL> @x

'BEFOREY'

------------
before y

'BEFOREZ'

------------
before z

'INZ'

--------
in z

'AFTERZ'

-----------
after z

'AFTERY'

-----------
after y

vertrace: wrote 7278 bytes to ~/.sqlcl/vertrace/traces/EBSCDB_ora_20751-3.trc
SQL> select * from dual;

DUMMY

--------
X
SQL>
```

4. Trace a script at a nested level (modifying y.sql to build upon example 3)

```
SQL> cat y.sql
vertrace next
select 'before z' from dual;
@z
select 'after z' from dual;
SQL> @x

'BEFOREY'

------------
before y

'BEFOREZ'

------------
before z

vertrace: wrote 4203 bytes to ~/.sqlcl/vertrace/traces/EBSCDB_ora_22198-1.trc

'INZ'

--------
in z

'AFTERZ'

-----------
after z

'AFTERY'

-----------
after y

SQL>
```

TRACEABLE SQLcl COMMANDS

There are three categories of commands that vertrace will trace. No other commands will trigger tracing.

SQL and PL/SQL

All SQL and PL/SQL commands produce database activity that ends up in an extended SQL tracefile.

Script execution

An experience is often contained in a script file (`.sql`). Therefore, the commands to execute a script are traceable at the highest nesting level where a tracing

directive occurred. A review of the examples will be helpful. The SQLcl script execution commands are:

```
@
@@
start
```

SQLcl

Some SQLcl commands (and their aliases) execute SQL and PL/SQL, including:

```
apex awr call ctas datapump(dp) describe(desc) execute(exec)
immutable_table(im) information(info) liquibase(lb) load
mle project(proj) sessions soda unload xquery
```

LOCAL TRACEFILE NOMENCLATURE

When tracefiles are saved locally to TRCDIR, you will notice the output names have a trailing sequence number that starts with 1 incrementing by 1 until a name is available. When you connect to Oracle (i.e., create an Oracle session), you should expect the sequence to start at 1 since the tracefile name will probably be unique with respect to your local trace directory. The only time you should see a new filename where the sequence is unexpected is when old files are left over that match the same name as the one that is being downloaded anew.

For example, assume your session's tracefile name on the database server is `EBSCDB_ora_4621.trc` and you trace two separate experiences. In your `TRCDIR`, you will find these tracefile names:

```
- EBSCDB_ora_4621-1.trc
- EBSCDB_ora_4621-2.trc
```

If, say, a year passes by without cleaning up old tracefiles, and you trace two separate experiences in a session whose tracefile shares the same name, then you should expect these new tracefile names:

```
- EBSCDB_ora_4621-3.trc
- EBSCDB_ora_4621-4.trc
```

COMMAND LINE PROCESSING

Command line completion and string quoting work like bash.

INSTALL

Prerequisites

- Method R Workbench 9.5.4.8 or later.
- SQLcl version 23.4 or later.
- Oracle 18c or later.

Note: Review the requirements on the Workbench installation page.

Note: With a minor modification to the database installation script Oracle 12c is supported.

Once per SQLcl location

Execute these steps for every installation of SQLcl where you want to use this extension.

1. Download the SQLcl distribution and install it if it isn't already
2. Download the distribution for this extension
3. Unzip the distribution in the lib/ext directory of the SQLcl home directory

Administrator instructions

This applies to anyone who needs to install the database objects or who needs to trace someone else's sessions.

Once per Administrator

Get installation scripts.

You can get them from a developer or you can execute the per SQLcl instructions.

You may modify only the code in the package bodies. And you may not modify the object names, schema name, type names, or role names. To do so would prevent the extension from working properly.

Once per database container

The installation script is currently designed to be installed in a pluggable and will probably require logging into a SYSDBA account to execute properly. Change directory to where the installation scripts are located and execute:

```
sql[plus] / as sysdba @xxjhvt_vertrace_install
```

It will first display all containers and then prompt you to enter a container name. Pick one, and then monitor the output for errors. If all is well, then grant the following role names according to your security policy:

```
XXJHVT_VERTRACE_DEVEL (if you need only to trace your own session)
XXJHVT_VERTRACE_ADMIN (if you need to trace some other session)
```

As written, the installations instructions require version 18c of the Oracle RDBMS. There are some modifications you can make to those instructions that would support a version as old as 7.0.16. Newer versions enable more features. The installation script has some comments regarding when certain commands were introduced.

Developer instructions

A developer executes these instructions only one time.

1. Download and install Method R Workbench
2. Execute the per SQLcl instructions
3. Execute `sqlcl /nolog` and then execute the stage command to expose the installation scripts
4. Have the administrator execute the administrative tasks

Of course, if the developer wants only raw trace data, then installing Method R Workbench is not required.

UNINSTALL

The stage subcommand also exposes a script that, when executed, will display commands that you can execute to remove everything that was installed in the database.

Afterward, you can uninstall the extension from your SQLcl home by removing all the files from its lib/ext folder matching the files within the vertrace distribution.

LIMITATIONS

This extension does not work when connecting to an instance with no accessible filesystem for storing tracefiles. This at least includes Oracle ADB-S and ADB-D. Even though there are views such as V$DIAG_TRACE_FILE_CONTENTS and SESSION_CLOUD_TRACE, querying them does not produce the same data that you would get from reading a corresponding file. Worse is the lifetime of their rows.

AUTHOR

Jeff Holt

VERSION

`vertrace` 1.11