TraceMiner2 Overview

Norman Dunbar

9th February 2017

Contents

TraceMiner2

An improved version of TraceMiner. Written in C++ and quite honestly, a better product altogether! Requires an Oracle 10g or upwards trace file, but can *probably* be changed to cope with 9i, at the very minimum, if necessary. Maybe. It depends.

Improvements Over TraceMiner.

It *may* not be faster, C++ never usually is, but it's a lot more thorough and safer too. It is less likely to bale out when it can't understand something in a trace file. Also:

- You don't have to reconfigure anything and recompile if you have wider (ie, more digits) cursor IDs on your system. It appears that the cursor IDs, since Oracle 10g, are the address in memory of the cursor, as opposed to a simple incrementing integer. I may be wrong.
- Likewise, there's no recompilation required if you have more than a certain number of bind variables in a statement.
- There is no longer a limit on the length of a bind variable's name.
- If the trace file has a line that is " value= Bind#n" then it is correctly coped with as two lines: a NULL bind value for the current bind variable, and the start of the following bind variable's data. Previously, you had to edit this line in the raw trace file. Not good!
- NULL values for bind variables are processed according to whether they are PL/SQL statement `OUT` variables. For PL/SQL `OUT` variables the variable's name is used as it's value - so you know it's an `OUT` value. For SQL statements, the bind's value must be `NULL`.
- If a bind variable is used in multiple places in the same SQL statement `update some_table set ... created_by = :username, creation_date = :today, last_updated_by = :username, last_update_date = :today, ... ;`, for example, then it is correctly reported in *each* of those places. TraceMiner, of old, didn't know what to do when that happened!
- If a bind is a ROWID type (oacdty=11 or 69) then the value line will have no quotes around the ROWID value. The output for the bind in the report will wrap the value in a `CHARTOROWID()` call - just to be explicit. If, however, the bind type is a simple `VARCHAR2`/`CHAR` there can be no wrapping, so there isn't!
- If a PL/SQL statement returns a ref_cursor (data type = 102), it's not possible to get a name or value for that particular bind as it simply points at an address in memory. In this situation, TraceMiner2 will replace the bind 'value' with the text "REF_CURSOR" - just so you know what's what.
- The log file and verbose output debugging file are created in the same directory as the trace file. No more messing about with redirection etc.
- The default report format is HTML. Plain text output is always going to be messy, as it replicates the input SQL statements, line feeds and all, which can be very messy indeed according to the developer or code generator that created them.
- The report file now reports the `EXEC`, `PARSE`, `BINDS` lines as well as the line in the trace file where the SQL was first located.

- `COMMIT` and/or `ROLLBACK` statements indicate whether or not data was actually changed. If you see a "COMMIT (Read Only)" in the report, you may be committing unnecessarily as the entire transaction made no changes.
- The report file repeats the headings every 25 EXECs by default. This makes the need to scrolling back and forth to find out what the numbers in the various columns actually refer to, a lot less necessary! This can be defined with a command line argument `--pagesize=nn` or `-p=nn`. No spaces are permitted around the '=' sign.
- A `TraceMiner2.css` file and a `favicon.ico` file are created in the same location as the trace file too. The CSS can be edited quite easily to match your installation standards - if you have any - or just to make things the way you prefer them. For some reason, Internet Explorer 11 completely ignores the icon file.
- If a css or icon file, as above, are detected in the trace file's folder, then they are not overwritten or changed in anyway. This way any changes you make are preserved.
- If running in verbose mode (`--verbose` or `-v`), the debugging output is much more readable, and much more verbose! If TraceMiner2 fails to parse a trace file, try running in verbose mode and see what gets output to the debug file.
- You can specify the maximum depth for cursors to be reported. The default is zero. (`--depth=n` or `-d=n`) Only cursors with a lower or equal depth will be reported. If you use `--depth=3` then only cursors with depth 0, 1 2 or 3 will be reported on.

Download the Source

Go to https://github.com/NormanDunbar/TraceMiner2 and click the clone or download button. Choose the option to download a zip file.

Save it somewhere safe, I use my `SourceCode` folder, and extract it. This will create a new folder named `TraceMiner2-master` - you can remove the `-master` bit if you wish, however, the following instructions assume that you didn't.

Compile the Source

On Linux

```
cd TraceMiner2-master
make -f makefile.gnu
```

On Windows

Visual C++/Visual Studio

Just don't! Trust me, you don't need the grief. It's not like TraceMiner2 uses anything non-standard, just plain old C++ and the Standard Template Library, for maximum portability. But then again, I did mention standards didn't I? That's a problem with Microsoft.

Honestly, try it if you want to. I haven't. TraceMiner of old needed so many changes to basic C code just to make it even think about compiling with Visual Studio, it's just not worth it.

Plus, I haven't even tried! :-)

Borland 10.1 compiler

```
cd TraceMiner2-master
bcc32c -o bin\TraceMiner2.exe TraceMiner2\*.cpp
```

CodeBlocks IDE

There is a project file in the `SourceCode\TraceMiner2-master\TraceMiner2` sub-folder, named `TraceMinder2.cbp`. Open that and select `Build->Build` or press CTRL-F9 to do the same. The executable will be found in `SourceCode\TraceMiner2-master\TraceMiner2\bin\Release` when it has completed.

Need a Free C++ Compiler?

https://www.embarcadero.com/free-tools is the place to look for one. It's 32 bit and modern. It runs perfectly well on 64 bit Windows too.

You will need to sign up, but other than a few special offers, and a couple of training course advisory emails, you won't get too much hassle. And it's worth it for one of the finest Windows C/C++ compilers for free.

- Download it.
- Unzip it.
- Add the bin folder to your path.
- Use it and love it! ;-)

Command Line Options

Executing TraceMiner2 is easy. At a minimum it requires a single parameter, the name of a trace file. This can be a relative or absolute path as desired. The trace file is expected to have a `.trc` extension.

```
TraceMiner2 [options] trace_file
```

Possible options are:

- `--help` or `-h` or `-?` which provides brief help about the application and its options, then exits without doing anything.
- `--pagesize=nn` or `-p=nn` which sets the report page size to `nn` EXEC statements. This throws a couple of blank lines and repeats the headings in a text format report, or, starts a new table with appropriate headings in an HTML report. In either case, it makes reading easier. The default is 25 and this works well using Firefox on Windows 7, or Linux with a decent sized monitor. Even Internet Explorer copes!

- `--text` or `-t` which forces the report file to be created in plain text mode. The default is to create the report in HTML format.
- `--verbose` or `-v` which creates a debugging file that will contain a huge amount of debugging information. If you have problems with TraceMiner2 then this file will help me debug things. It's best, really, that you don't run the application in this mode unless absolutely necessary! You have been warned. :-)
- `--quiet` or `-q` will turn off all the `Cursor: #cccccc created at line nnnn` messages. Any `ERROR #ccccc` or `PARSE ERROR #cccccc` lines will still be reported though. You can't turn those off.
- `--depth=n` or `-d=n` which determines which cursors will be examined and reported on. Any cursor with a `PARSE`, `EXEC` or `CLOSE`, with a `dep=` value less than or equal to the supplied `--depth=n` will be reported. Anything with a `dep=` value greater than the requested depth will be ignored.

Traceminer2 will create:

- A report file, the default is in HTML format, which is the same name as the trace file, but with the extension changed from `.trc` to `.html`.
- If the report is in HTML format, then `favicon.ico` will be created, *if one doesn't already exist* in the folder the trace file is found in.
- If the report is in HTML format, then `TraceMiner2.css` will be created, *if one doesn't already exist* in the folder the trace file is found in. This file allows you to style the HTML report as per your company standards (well, up to a point) or to your preference.
- A debugging file. If and only if running in verbose mode. This will have the same name as the trace file used for input, but with the extension changed from `.trc` to `.dbg`.

Testing

Trace Miner 2 has been tested on the following:

Compiled using Borland C++ 10.1:

- Windows 7.
- Windows Server 2012.

Compiled with G++ 4.8.5:

- Oracle Enterprise Linux 7.2
- Oracle Enterprise Linux 7.3
- CentOS Enterprise Linux 7.2

Please note: TraceMiner2 tries to use the Standard Template Library's REGEX facilities. However, these do not work in versions of the STL supplied with `g++` at versions less than 4.9.0. there is a check in the source code (in `gnu.h`) to determine if the compiler works or doesn't, and if not, uses plain old scanning for the required data - such as cursor IDs, depth, lengths of SQL statements etc.

Compiled with G++ 5.x.x:

- Linux Mint 18.

Documentation

In development, `doxygen` version 1.8.13 was used. Versions previous to this *may* cause problems. Time and testing will tell.

The source code is documented with specially formatted comments which are collected by *Doxygen* http://www.stack.nl/~dimitri/doxygen/, and output as HTML files documenting all the source and header files, classes, variables, functions etc in the source code. Other output formats are available, but HTML is the default.

There is a Doxygen configuration file included in the source code. It can be found in the folder `SourceCode\TraceMiner2-master\Docs` and is named `TraceMiner2.doxyfile`.

Sadly this file doesn't appear to enable relative paths to be used when extracting from the source files, so if you desire to generate the documentation, you will have to edit the document in a number of places:

Option	Current Value
PROJECT_LOGO	C:/SourceCode/TraceMiner2/Docs/tm2_logo_25pct.png
OUTPUT_DIRECTORY	C:/SourceCode/TraceMiner2/TraceMiner2/doxygen
WARN_LOGFILE	C:/SourceCode/TraceMiner2/TraceMiner2/doxygen/doxygen.log
INPUT	C:/SourceCode/TraceMiner2/TraceMiner2/
HAVE_DOT	YES

Replace `C:/SourceCode/TraceMiner2` with your appropriate top level directory.

The configuration file for Doxygen assumes the presence of the `dot` (aka `Graphviz` http://www.graphviz.org/) utility to draw the class, call and caller diagrams. If you don't have dot, or do not wish to generate these diagrams, then set `HAVE_DOT` to `NO`.

Once you have edited - and saved - the configuration file, run it as follows (assuming you have Doxygen installed and on your path):

```
doxygen /path/to/config/file
```

for example:

```
doxygen C:/SourceCode/TraceMiner2/Docs/TraceMiner2.doxyfile
```

The HTML folder, found in the location you set for `OUTPUT_DIRECTORY` above, will contain the `index.html` file that starts the documentation proper.

If you have installed Doxygen, then there is a Wizard that you can run to load the configuration file and generate the documentation. On Windows 7, it is found under `start->all programs->doxygen->Doxywizard`.

Figure 1: The Trace Miner!

Appendices

The following appendices contain potentially useful information about the lines of the trace file that are parsed.

Appendix A - Trace File changes known to prevent 9i being used.

From 10g onwards, Oracle changed the format of some of the lines in a trace file. At present, one that is known to me is relating to the data type of a bind variable. (In the `Bind#n` section). In 9i this was simply `dty` but from 10g it changed to `oacdty`. Currently TraceMiner2 (and indeed TraceMiner of old), doesn't check for this value.

Appendix B - Oracle Command Codes

The `oct` parameter in a PARSING IN CURSOR line in an Oracle trace file, determines the command that is being parsed in the SQL statement.

Known command types are:

Command Code	Data Type
2	INSERT
3	SELECT
6	UPDATE
7	DELETE
26	LOCK TABLE
44	COMMIT
45	ROLLBACK
46	SAVEPOINT
47	PL/SQL Block
48	SET TRANSACTION
55	SET ROLE
90	SET CONSTRAINTS
170	CALL
189	MERGE

At the moment, TraceMiner2 only considers the command type when it comes across a bind variable that is either a PL/SQL OUT parameter, or, a SQL statement where a NULL is being passed in (or out!). It uses the command code to determine which to substitute for the bind variable name in the statement.

Appendix C - Bind Data Types

The `oacdty` parameter in a bind variables details determines the data type of that bind variable. This is not necessarily the data type of the column in a table that it may be being `INSERT`ed or `UPDATE`d into, or compared against.

The following data are taken from the *Internal Data Types* section of the *Data Types* chapter in the 11gR2 *Oracle Call Interface* manual, which you can find at http://docs.oracle.com/cd/B19306_01/appdev.102/b14250/oci03typ.htm.

Listed data types are:

Data Code	Data Type
1	VARCHAR2 or NVARCHAR2
2	NUMBER
8	LONG
11	ROWID
12	DATE
23	RAW
24	LONG RAW
25	Unhandled data type
29	Unhandled data type
69	ROWID
96	CHAR or NCHAR
100	BINARY_FLOAT
101	BINARY_DOUBLE
102	REF_CURSOR
108	User-defined type (object type, VARRAY, nested table)
111	REF
112	CLOB or NCLOB
113	BLOB
114	BFILE
123	VARRAY
180	TIMESTAMP
181	TIMESTAMP WITH TIME ZONE
182	INTERVAL YEAR TO MONTH
183	INTERVAL DAY TO SECOND
208	UROWID
231	TIMESTAMP WITH LOCAL TIME ZONE

However, various other sources on the internet seem to disagree with what the above table shows. In addition, I have at least one Oracle Trace where a ROWID is type 11 and not type 69, also, I have VARRAY as type 123 and not as type 108. Consistency? Who mentioned consistency?

TraceMiner2 has a simple manner of extracting the bind variable value, as briefly explained below.

VARCHAR2 and NVARCHAR2

If the data type is 1, and the value is wrapped in double quotes then the data value is simply the string between the quotes. TraceMiner2 replaces the double quotes with single quotes to match those that the original SQL statement probably used.

If the first character of a type 1 data bind is *not* a double quote, then the remainder of the trace line holds hex pairs in the format `0 30 0 31 0 32 0 33` etc (for '0123' in this example) and the data type is `NVARCHAR2`.

Examples

```
BIND#0
...
value="DELETE ME PLEASE"
...
BIND#1
...
value=0 30 0 31 0 32 0 32 0 64 0 65 ...
...
```

In the latter case, '0122AB', TraceMiner2 only extracts the data bytes from the *first* line of the text. Continuation lines are simply, at present, ignored. There can be up to 50 bytes of data on the first line, and this is considered adequate. Unless, of course, you know better!

NUMBER

For data type 2, the bind's value is just the string of digits extracted from the remainder of the trace line.

One thing to watch for, if the value of the numeric bind is "###" in the trace file, then this is most likely a PL/SQL OUT parameter, and as such, TraceMiner substitutes the bind's name rather than "###".

Examples

```
BIND#0
...
value=1357666
...
```

CHAR or NCHAR

These are similar to VARCHAR2 and NVARCHAR2 above and indeed are processed by the same code.

ROWID

If a bind variable's type (`oacdty=11` or `oacdty=69`) then the value in the trace file will not be wrapped in quotes. TraceMiner2 outputs such binds wrapped in a call to `CHARTOROWID()` to make the fact that this is a `ROWID` variable, and not a character variable, explicit.

LONG, RAW and LONG RAW

At present, and because mainly I have no test data, these data types are not catered for. I *suspect* it might be a string of hex digits, space separated perhaps, but I don't yet know. When I do, I'll update TraceMiner2.

VARRAY

At present, I've only seen these in the array buffer that receives the lines of text from `DBMS_OUTPUT.GET_LINES()` call. In this case, the value is missing, so we simply substitute the array name again. It's not perfect, but it's better than nothing (or `NULL`!)

REF_CURSOR

If a bind variable has data type 102, then it's almost certainly a `PL/SQL OUT` parameter defining a `REF_CURSOR`. As it is not possible to extract a value - the trace file simply dumps out an address in memory for these types - TraceMiner2 will indicate the bind's usage by substituting the text "REF_CURSOR" into the statement.

The following is a brief example of what I mean above, it shows a `PL/SQL OUT` bind for a `REF_CURSOR`:

```
 Bind#3
  oacdty=102 mxl=04(04) mxlc=00 mal=00 scl=00 pre=00
  oacflg=01 fl2=1000000 frm=00 csi=00 siz=0 off=176
  kxsbbbfp=c2a91548  bln=04  avl=04  flg=01
  value=Unhandled datatype (102) found in kxsbndinf
Dump of memory from 0x00000000C2A91548 to 0x00000000C2A9154C
0C2A91540                  00000000                  [....]
```

Others.

Any data type not mentioned above are simply considered to have the value of whatever is on the remainder of the trace file line after the text `value=`.

Appendix D - Bind Variable Details

The following are some of the fields used in the descriptions of each individual Bind variable in a trace file. Anything not listed below is of an unknown nature.

Code	Description
oacdty	Data type code.
mxl	Maximum length of the bind variable value.
mal	Array length.
scl	Scale - NUMBER data types only (oacdty = 2).
pre	Precision - NUMBER data types only (oacdty = 2).
oacflg	Special flag indicating bind options.
fl2	Second part of oacflg.
csi	Character set identifier. (See Appendix D)
siz	Amount of memory to be allocated for this chunk.
off	Offset into the chunk of the bind buffer.
kxsbbbfp	Bind address.
bln	Bind buffer length.

Code	Description
avl	Actual value length.
flg	Bind status flag.
value	Value of the bind variable (See Appendix C).

For some reason, TraceMiner2 refers to `avl` as *average* length, not *actual*. Sigh.

Appendix E - Character Set Codes

Some data types use different character sets. These are coded in the `csi` field as listed above. Typical values that you may see here are as follows:

Code	Character Set
1	US7ASCII
31	WE8ISO8859P1
46	WE8ISO8859P15
170	EE8MSWIN1250
178	WE8MSWIN1252
871	UTF8
873	AL32UTF8
2000	AL16UTF16

Examples

```
Bind#0
 oacdty=96 mxl=128(50) mxlc=00 mal=00 scl=00 pre=00
 oacflg=01 fl2=1000010 frm=02 csi=2000 siz=128 off=0
 kxsbbbfp=1109ffe98  bln=128  avl=22  flg=05
 value=0 34 0 32 0 35 0 33 0 35 0 32 0 2d 0 39 0 30 0 30 0 37
```

This bind has `csi=2000` so it is using the ALUTF16 character set for its value, which happens to decode as '425352-9007'.

```
Bind#1
 oacdty=01 mxl=32(04) mxlc=00 mal=00 scl=00 pre=00
 oacflg=10 fl2=0001 frm=01 csi=31 siz=0 off=24
 kxsbbbfp=610cd550  bln=32  avl=04  flg=01
 value="DUAL"
```

This bind, on the other hand, has `csi=31` so it is using the WE8ISO8859P1 character set. You can see the value in the extract above.

Appendix F - Credits

Image Credits

The miner image used in TraceMiner's logo - which you can see when you generate the documentation - is used with gratitude. I obtained it from http://www.clipartpanda.com/categories/miner-clipart.

Pandoc

The PDF version of the `README.md` (markdown) file was created using `pandoc`, which you can obtain and use for free from http://pandoc.org/. It will convert almost any source file into almost any output file.

The PDF was created thus:

```
export PANDOC_PDF=README.pdf
export PANDOC_COLOUR="Cool Grey"

pandoc --from markdown \
--to latex \
--output "${PANDOC_PDF}" \
--table-of-contents \
--toc-depth=3 \
--listings \
--include-in-header listings_setup.tex \
--variable fontfamily="utopia" \
--variable toccolor="${PANDOC_COLOUR}" \
--variable linkcolor="${PANDOC_COLOUR}" \
--variable urlcolor="${PANDOC_COLOUR}" \
--variable margin-top=3cm \
--variable margin-left=3cm \
--variable margin-right=3cm \
--variable margin-bottom=4cm \
README.md
```

Other, useful, output formats are `epub`, Word `docx`, Libre Office `odt`, `ReStructuredText`, `HTML` etc etc etc.