



ubTools

ubSQL 6.0



Introduction

- What is ubSQL ?
- Unload architecture
- Running ubSQL
- Commands
- Parameters
- Unload performance tips



What is ubSQL ?

- Command line application.
- Unloads Oracle data to text and binary files.
- Runs DML,DDL statements.
- Easy to port to different platforms.



Supported Objects

- Any objects that work within a SELECT statement.

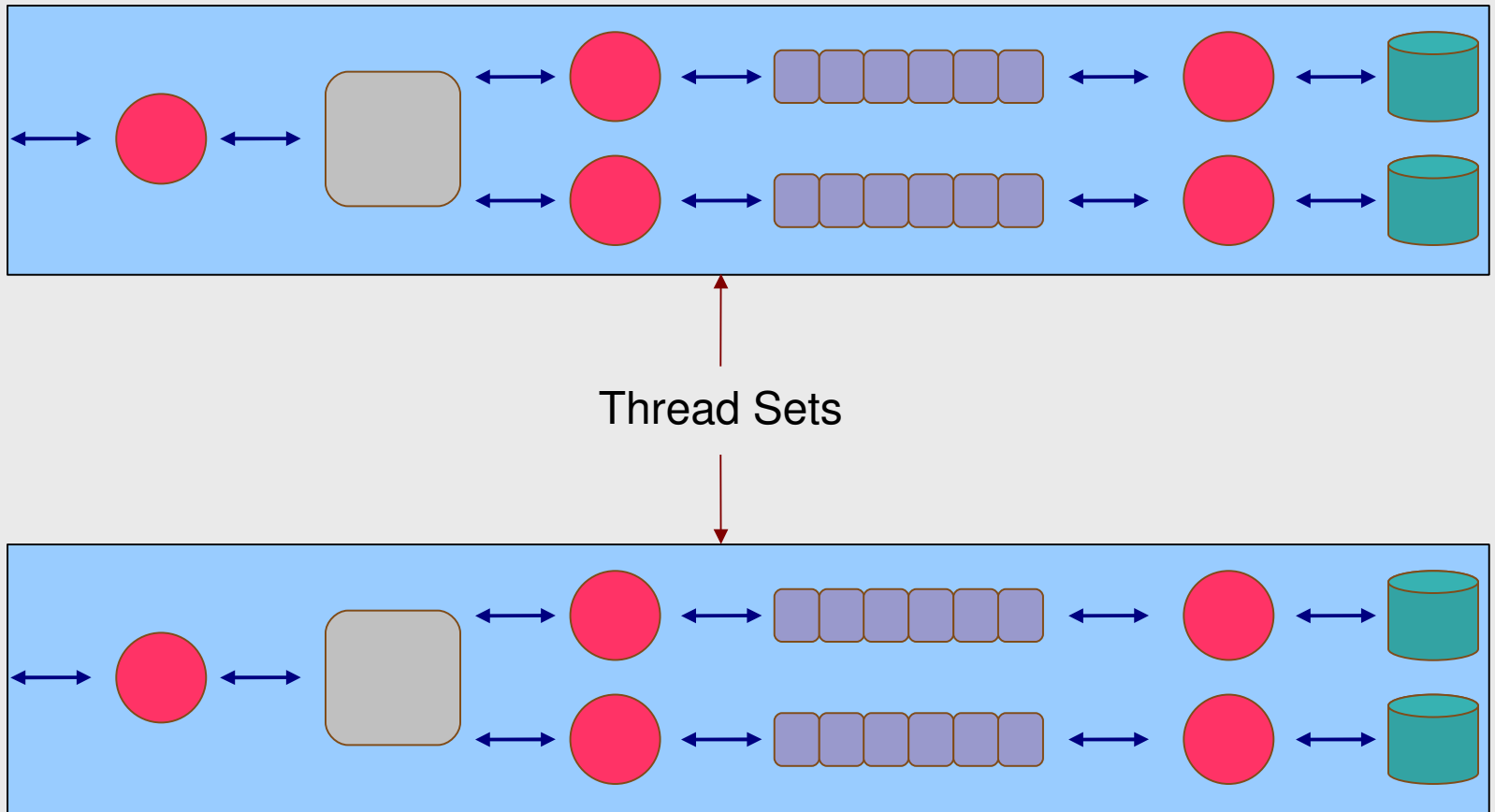
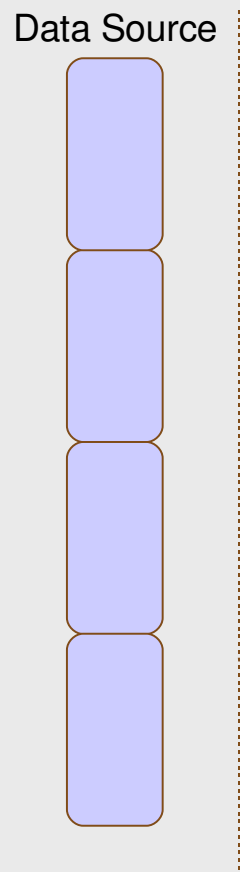


Supported Data Types

- All Numeric data types
- CHAR, VARCHAR2
- DATE
- RAW
- TIMESTAMP, TIMESTAMP WITH TIME ZONE
- INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND

Unload Architecture

fetchThreads inputBuffers fillerThreads outputBufferSets writerThreads outputFiles





Running ubSQL

Interactive:

```
$ ./ubSQL

ubSQL: Release 6.0.0 - Production - _POSIX_VERSION: 200112

Copyright (c) 2005, ubTools, Adana/Turkey. All rights reserved.

Not connected to an instance.

ubSQL>
```

Script:

```
$ ./ubSQL < ../output/ubsql_test.sql
.....
$
```

Script and Log:

```
$ ./ubSQL < ../ubsql_test.sql > ../ubsql_test.log
$
```



Commands

- encode ...;
- connect <username>/<password>@<tnsAlias>;
- show parameters;
- set <paramName>=<paramValues>;
- unload <any syntax used in SELECT statement>;
- create sql_loader control_file ...;
- create oracle_loader ddl_file ...;
- exit;
- Any Oracle DML,DDL statements.



Command: ENCODE

```
ubSQL> encode 'password123';  
MC8zLzEtMy00LzAvMS1xYHJydm5zZTAzMg==  
Encoded.  
ubSQL>
```



Command: CONNECT

```
ubSQL> connect dunal/password123@dunal;
```

```
Connected.
```

```
ubSQL>
```



Command: CONNECT encoded

```
ubSQL> encode 'password123';
```

```
MC8zLzEtMy00LzAvMS1xYHJydm5zZTAzMg==
```

```
Encoded.
```

```
ubSQL> connect
```

```
dunal/e'MC8zLzEtMy00LzAvMS1xYHJydm5zZTAzMg=='@dunal;
```

```
Connected.
```

```
ubSQL>
```



Command: SHOW

```
ubSQL> show parameters;
```

NAME	MODIFIABLE	VALUE
ARRAY_FETCH_ROW_COUNT	TRUE	1000
BYPASS_FILE_SYSTEM_CACHE	TRUE	FALSE
COLUMN_DELIMITER	TRUE	
EXIT_ON_ERROR	FALSE	FALSE
FEEDBACK_ROWS	TRUE	1000000
LINE_DELIMITER	TRUE	x'0A'
LOG_FILE_LEVEL	FALSE	3
LOG_FILE_NAME	FALSE	ubSQL.log
NLS_EXPANSION_FACTOR	TRUE	1
OUTPUT_BUFFERS_PER_FILE	TRUE	100
OUTPUT_BUFFER_SIZE	TRUE	65536
OUTPUT_FILE_FORMAT	TRUE	TEXT
OUTPUT_FILE_NAMES	TRUE	../output/4.dat
SESSION_PARAMETERS	TRUE	
nls_date_format=RRDDSSSSS	"_serial_direct_read"	=true

```
ubSQL>
```

LICENSE_KEY and internal parameters are not printed.



Command: SET

```
ubSQL> SET OUTPUT_FILE_NAMES=a.txt,b.txt,c.txt;
```

```
Parameter altered.
```

```
ubSQL> set SESSION_PARAMETERS=nls_date_format=RRDDSSSSS  
timed_statistics=true;
```

```
Parameter altered.
```

```
ubSQL> set LINE_DELIMITER=X'0A';
```

```
Parameter altered.
```

```
ubSQL>
```



Command: UNLOAD

```
ubSQL> unload a.* from dunal.ubsql_test a;

1000000 rows fetched by FetchThread 0.

1060247 rows unloaded by ThreadSet 0.

ThreadSet:0, FetchThread,      Elapsed Time:15.21 (Filler Thread
Wait Time:0.00 Database+Network+OCI+Other Wait Time:14.92)
ThreadSet:0, FillerThread:0, Elapsed Time:1.95 (Writer Thread Wait
Time:0.01)
ThreadSet:0, WriterThread:0, Elapsed Time:2.64

Service time...: 11.60
Elapsed time...: 15.35

ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

- Definition: Number of parallel degrees and split level for object.
- Implementation: Product of parallel degrees thread sets are created.

The following statement creates 8 thread sets.

```
UNLOAD /*+ UB_PARALLEL(a,2) UB_PARALLEL(b,4) */ ...
```

Command: UNLOAD

UB_PARALLEL hint

- Object split levels:

TABLE_SPLIT: Split operation is done in table level.

PARTITION_SPLIT: If possible, split operation is done in partition level.

SUBPARTITION_SPLIT: If possible, split operation is done in subpartition level.

Split level used table:

HINT	OBJECT TYPE		
	<u>TABLE</u>	<u>PARTITION</u>	<u>SUBPARTITION</u>
<u>TABLE_SPLIT</u>	table_split	table_split	table_split
<u>PARTITION_SPLIT</u>	table_split	partition_split	partition_split
<u>SUBPARTITION_SPLIT</u>	table_split	partition_split	subpartition_split

Command: UNLOAD

UB_PARALLEL hint

SYNTAX:

```
UNLOAD [ /*+ {UB_PARALLEL(<objectName>,  
                                <parallelDegree>  
                                [ ,TABLE_SPLIT |  
                                PARTITION_SPLIT |  
                                SUBPARTITION_SPLIT  
                                ]  
                                )  
        }  
        */  
]
```

<u>Token</u>	<u>Description</u>	<u>Optional</u>	<u>Default</u>
objectName	Data source name or alias name	No	
parallelDegree	Degree of parallelism	No	
	Object split levels	Yes	TABLE_SPLIT

Command: UNLOAD

UB_PARALLEL hint

```
ubSQL> unload /*+ ub_parallel(a,2) */ a.* from dunal.ubsql_test a;

Finding object split ranges...
UB_PARALLEL object.....: a
Split level requested...: TABLE_SPLIT
Split level used.....: TABLE_SPLIT
Object split ranges found.

475645 rows unloaded by ThreadSet 1.

ThreadSet:1, FetchThread,      Elapsed Time:4.87 (Filler Thread Wait Time:0.90
Database+Network+OCI+Other Wait Time:3.96)
ThreadSet:1, FillerThread:0, Elapsed Time:4.42 (Writer Thread Wait Time:0.00)
ThreadSet:1, WriterThread:0, Elapsed Time:0.27

584602 rows unloaded by ThreadSet 0.

ThreadSet:0, FetchThread,      Elapsed Time:5.98 (Filler Thread Wait Time:1.28
Database+Network+OCI+Other Wait Time:4.68)
ThreadSet:0, FillerThread:0, Elapsed Time:5.19 (Writer Thread Wait Time:0.00)
ThreadSet:0, WriterThread:0, Elapsed Time:0.42

Service time...: 9.48
Elapsed time...: 9.93

ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

```
ubSQL> unload /*+ ub_parallel(a,2,table_split) */ a.* from dunal.ubsql_test a;

Finding object split ranges...
UB_PARALLEL object.....: a
Split level requested...: TABLE_SPLIT
Split level used.....: TABLE_SPLIT
Object split ranges found.

475645 rows unloaded by ThreadSet 1.

ThreadSet:1, FetchThread,      Elapsed Time:5.36 (Filler Thread Wait Time:0.61
Database+Network+OCI+Other Wait Time:3.96)
ThreadSet:1, FillerThread:0, Elapsed Time:1.31 (Writer Thread Wait Time:0.62)
ThreadSet:1, WriterThread:0, Elapsed Time:1.10

584602 rows unloaded by ThreadSet 0.

ThreadSet:0, FetchThread,      Elapsed Time:6.81 (Filler Thread Wait Time:0.72
Database+Network+OCI+Other Wait Time:4.84)
ThreadSet:0, FillerThread:0, Elapsed Time:1.88 (Writer Thread Wait Time:0.66)
ThreadSet:0, WriterThread:0, Elapsed Time:1.19

Service time...: 9.41
Elapsed time...: 9.83

ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

- Requirements:

SELECT ON DBA_OBJECTS, DBA_SEGMENTS and DBA_EXTENTS privileges are required.

Without these privileges:

```
ubSQL> unload /*+ ub_parallel(a,2) */ a.* from dunal.ubsql_test;  
  
ORA-00942: table or view does not exist  
  
Service time...: 0.01  
Elapsed time...: 4.25  
  
ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

- Limitations:

Aggregate functions(COUNT,SUM,etc.) and ROWNUM on returned rows are not supported. This would cause wrong results since each thread set processes its own rows. This limitation does not exist in subqueries.

```
ubSQL> unload /*+ ub_parallel(a,2) */ count(*) from dunal.ubsql_test;  
  
ORA-02014: cannot select FOR UPDATE from view with DISTINCT, GROUP BY, etc.  
  
Service time...: 0.00  
Elapsed time...: 0.11  
  
ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

- Limitations:

UB_PARALLEL hint is supported only in hint clause which starts immediately after UNLOAD keyword. Any other UB_PARALLEL hints are ignored.

```
ubSQL> unload /* comment */ /*+ ub_parallel(a,2) */ a.* from ubsql_test a;

.....
1060247 rows unloaded by ThreadSet 0.
.....
ubSQL> unload a.* from ubsql_test a where a.f1 in (select /*+ ub_parallel(b,2) */
b.f1 from ubsql_test2 b);

.....
22 rows unloaded by ThreadSet 0.
.....
ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

Limitations:

Unloading all columns by '*' is not supported. Use "<aliasName>.*" as a workaround.

```
ubSQL> unload /*+ ub_parallel(a,2) */ * from ubsql_test a;
Finding object split ranges...
ORA-00936: missing expression

Service time...: 0.00
Elapsed time...: 0.01
ubSQL> unload /*+ ub_parallel(a,2) */ a.* from ubsql_test a;
.....
584602 rows unloaded by ThreadSet 0.
.....
ubSQL>
```

Command: UNLOAD

UB_PARALLEL hint

Limitations:

Unloading same column names of different objects is not supported. Use alias name as a workaround.

```
ubSQL> unload /*+ ub_parallel(a,2) */ a.f1, b.f1 from dunal.ubsql_test a,  
dunal.ubsql_test b where a.f1=b.f1;  
ORA-00918: column ambiguously defined  
  
Service time...: 0.00  
Elapsed time...: 0.05  
ubSQL> unload /*+ ub_parallel(a,2) */ a.f1 a_f1, b.f1 b_f1 from dunal.ubsql_test a,  
dunal.ubsql_test b where a.f1=b.f1;  
.....  
1000000 rows fetched by FetchThread 0.  
.....  
ubSQL>
```

Command: CREATE SQL_LOADER

SYNTAX:

```
CREATE SQL_LOADER CONTROL_FILE=<controlFileName>  
  PARAMETERS="<sqlLoaderParameters>"  
  FOR "<SQL>"
```

<u>Clause</u>	<u>Description</u>	<u>Optional</u>	<u>Default</u>
CONTROL_FILE	Control file name	No	
PARAMETERS	Oracle dependent SQL*Loader parameters	No	
FOR "..."	SQL including column description	No	

Command: CREATE SQL_LOADER

```
ubSQL> create sql_loader control_file=../output/sil0.ctl
parameters="
options(direct=true)
load data
infile '../output/test0.dat'
into table dunal.sil3 append"
for select * from dunal.sil0;

CREATE completed.

ubSQL>
```

Command: CREATE ORACLE_LOADER

SYNTAX:

```
CREATE ORACLE_LOADER DDL_FILE=<scriptName>
  EXTERNAL_TABLE=<externalTableName>
  DEFAULT_DIRECTORY=<defaultDirectoryName>
  UNLOAD_SQL="<unloadSQL>"
  [ ACCESS_PARAMETERS="<accessParameters>" ]
  [ LOCATION=<outputFileName{,<outputFileName>} ]
  [ LOAD_SQL="<loadSQL>" ]
  [ REJECT_LIMIT=<rejectLimitCount> ]
  [ TABLE_PROPERTIES="<tableProperties>" ]
```

<u>Clause</u>	<u>Description</u>	<u>Optional</u>	<u>Default</u>
DDL_FILE	External table creation script name	No	
EXTERNAL_TABLE	External table name	No	
DEFAULT_DIRECTORY	External table default directory name	No	
UNLOAD_SQL	SQL including column description	No	
ACCESS_PARAMETERS	Oracle dependent external table access parameters	Yes	
LOCATION	Output file names	Yes	OUTPUT_FILE_NAMES
LOAD_SQL	SQL to load external table	Yes	
REJECT_LIMIT	Oracle dependent external table reject limit	Yes	
TABLE_PROPERTIES	Oracle dependent table properties	Yes	

Command: CREATE ORACLE_LOADER

```
ubSQL> create oracle_loader ddl_file=../output/sil0.sql
external_table=dunal.sil0_ext default_directory=ext_tab_dir
unload_sql ="select * from dunal.sil0";
```

CREATE completed.

```
ubSQL> create oracle_loader ddl_file=../output/sil0.sql
external_table=dunal.sil0_ext default_directory=ext_tab_dir
unload_sql ="select /*+ ub_parallel(a,2) */ * from dunal.sil0 a";
```

CREATE completed.

```
ubSQL> create oracle_loader ddl_file=../output/sil0.sql
external_table=dunal.sil0_ext default_directory=ext_tab_dir
unload_sql ="select * from dunal.sil0"
access_parameters="DATE_CACHE 1000"
location=test0.dat,test1.dat
load_sql="insert /*+ append */ into dunal.sil3"
reject_limit=0 table_properties="PARALLEL 4";
```

CREATE completed.

```
ubSQL>
```



Commands: Oracle DML/DDL

```
ubSQL> insert into dunal.sil3 values(1,'dunal');
```

```
1 rows inserted.
```

```
ubSQL> drop table dunal.sil3;
```

```
DROP completed.
```

```
ubSQL>
```



Command: EXIT

```
ubSQL> exit;
```

```
$
```

Parameter:

ARRAY_FETCH_ROW_COUNT

- Definition: Number of rows per FETCH.
- Implementation: Number of ARRAY_FETCH_ROW_COUNT returned to array per FETCH.

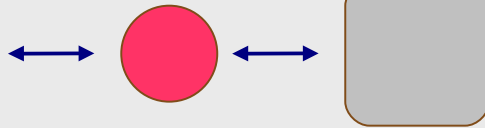
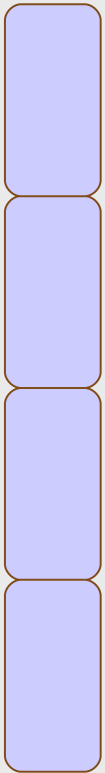
Parameter:

ARRAY_FETCH_ROW_COUNT

fetchThreads

inputBuffers

Data Source



$ARRAY_FETCH_ROW_COUNT = \text{Rows per fetch into each inputBuffer.}$

Parameter:

BYPASS_FILE_SYSTEM_CACHE

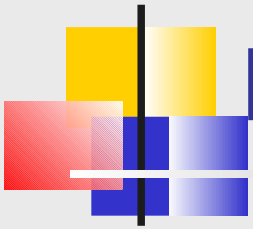
- Definition: Enables DIRECT_IO on write() system call to output files.
- Implementation:

TRUE: Data in output buffers are written directly to disk without copying to file system cache.

FALSE: Data in output buffers are copied to file system cache first, then written to disk.

As of this version, this parameter is implemented only in Linux.

- Recommendation: This parameter may be removed in a future release. Set it to FALSE. When it's FALSE, writing to file system cache is optimized by ubSQL and other users' useful data are not flushed out from file system cache.



Parameter: COLUMN_DELIMITER

- Definition: Column delimiter.
- Implementation: It can be a single character or a string. HEXADECIMAL values are not accepted.



Parameter: EXIT_ON_ERROR

- Definition: Exit behaviour whenever an error occurred
- Implementation: If EXIT_ON_ERROR=TRUE, whenever an error occurs, application exits. If EXIT_ON_ERROR=FALSE, application exits only an internal error occurs.

If application exits successfully, it returns 0; otherwise it returns a non zero value.



Parameter: FEEDBACK_ROWS

- Definition: Number of rows giving feedback about unload operation.
- Implementation: After each FEEDBACK_ROWS fetched, a message to screen is printed to give feedback.



Parameter: LINE_DELIMITER

- Definition: Line delimiter.
- Implementation: It must be a single character. HEXADECIMAL values are accepted if it is enclosed in x'' or X'', such as X'0A'.



Parameter: LOG_FILE_LEVEL

- Definition: Log file level.
- Implementation: It's not used in this version.



Parameter: LOG_FILE_NAME

- Definition: Log file.
- Implementation: It's created but not written in this version.



Parameter:

NLS_EXPANSION_FACTOR

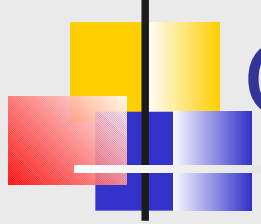
- Definition: NLS expansion factor of character types.
- Implementation: Byte size of character types in database is multiplied by NLS_EXPANSION_FACTOR to allocate large enough buffers in client, in case of buffer expansion during multibyte NLS conversion between client and database .

Parameter:

OUTPUT_BUFFERS_PER_FILE

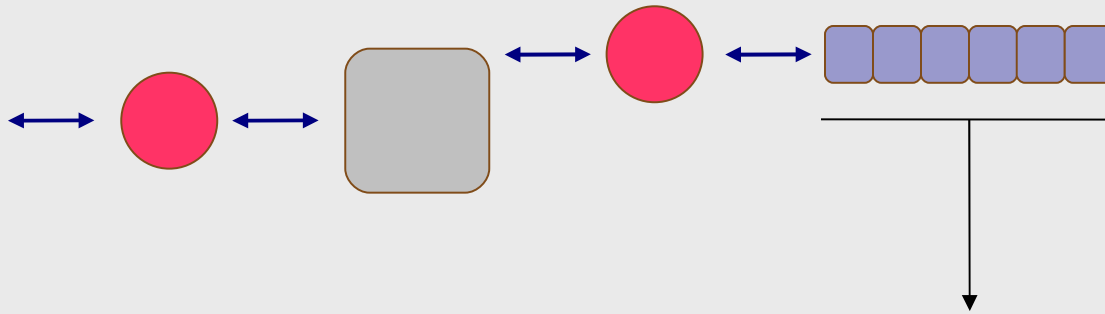
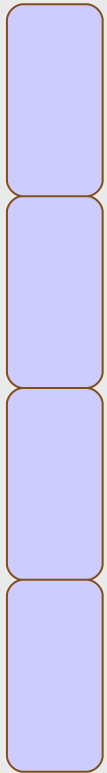
- Definition: Number of output buffers per output file.
- Implementation: If output buffer is filled, next output buffer is waited until it's free.

Parameter: OUTPUT_BUFFERS_PER_FILE



fetchThreads inputBuffers fillerThreads outputBufferSets

Data Source



OUTPUT_BUFFERS_PER_FILE

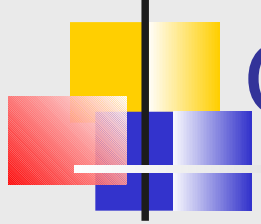


Parameter:

OUTPUT_BUFFER_SIZE

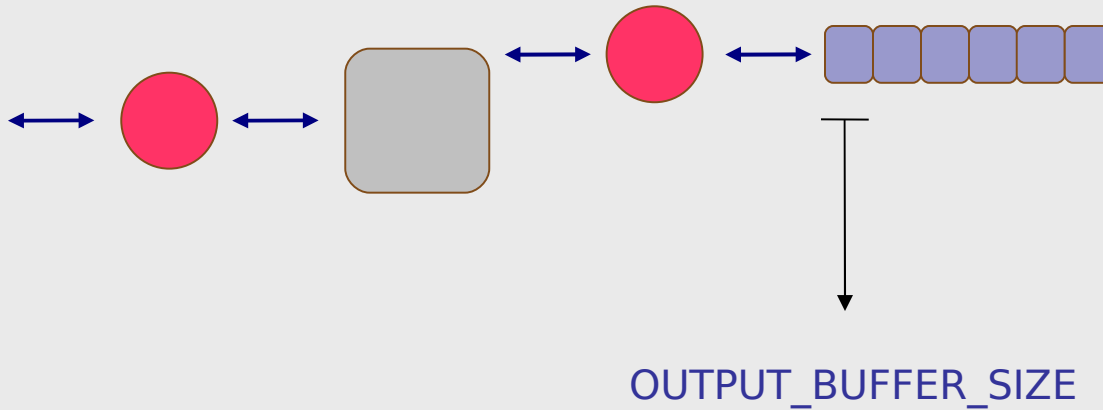
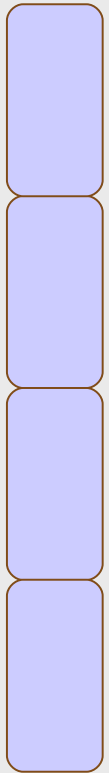
- Definition: Number of bytes of each output buffer.
- Implementation: If output buffer is filled, output buffer is written to output file.

Parameter: OUTPUT_BUFFER_SIZE



fetchThreads inputBuffers fillerThreads outputBufferSets

Data Source



$$\begin{aligned}
 \text{totalOutputBufferSize} &= (\text{thread sets}) \\
 &\quad * \text{OUTPUT_BUFFERS_PER_FILE} \\
 &\quad * \text{OUTPUT_BUFFER_SIZE} \\
 &\quad * \text{fileCount}(\text{OUTPUT_FILE_NAMES})
 \end{aligned}$$



Parameter: OUTPUT_FILE_FORMAT

- Definition: Output file format.
- Implementation:

TEXT: Output file format is text format. It can be loaded to non-Oracle systems, and also to Oracle by SQL*Loader, external table.

BINARY: Output format is binary. It can be loaded to Oracle database by external table.

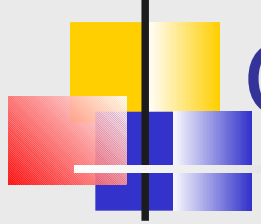


Parameter: OUTPUT_FILE_NAMES

- Definition: Output file names.
- Implementation: Each file has its own buffers grouped by OUTPUT_BUFFERS_PER_FILE that is called Buffer Set.

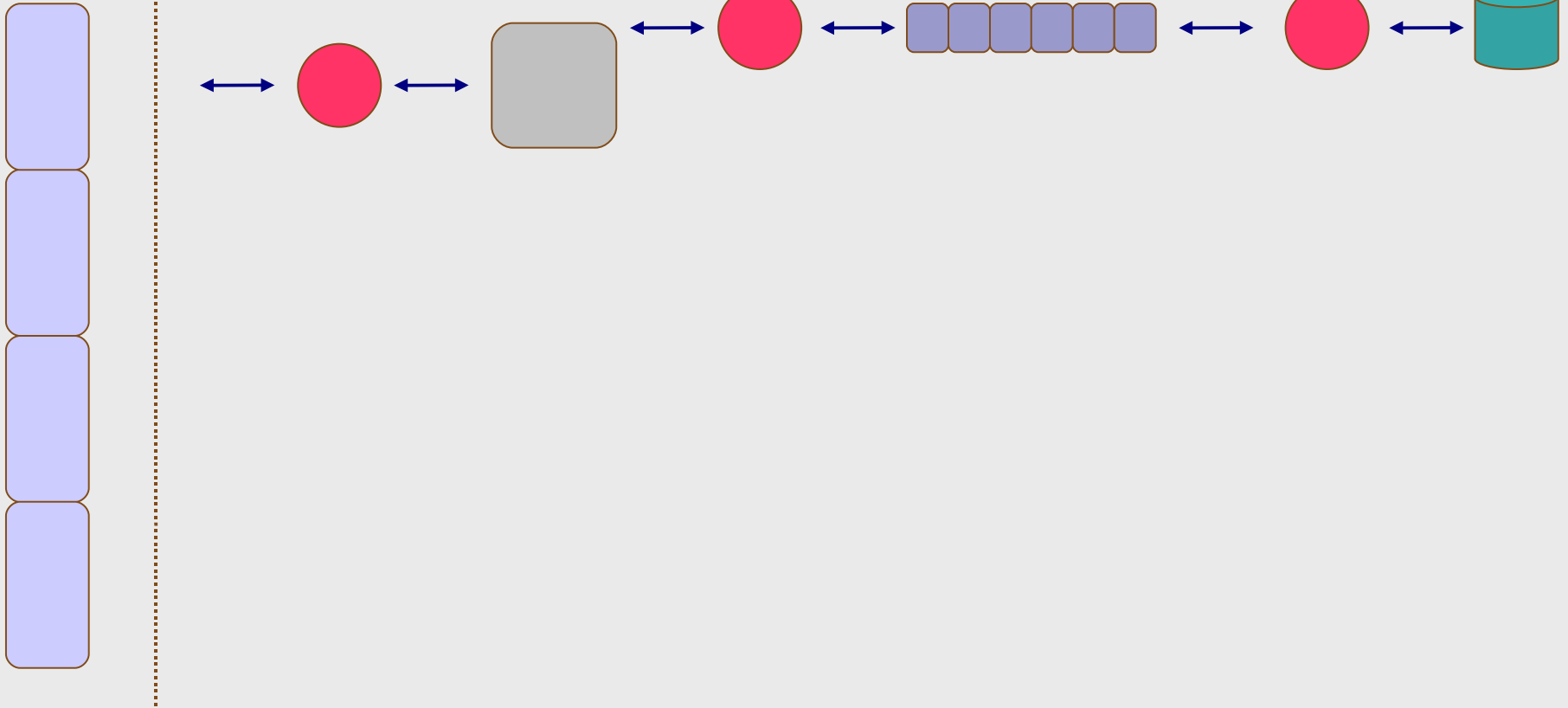
File names are appended by thread set id if UB_PARALLEL hint is used.

Parameter: OUTPUT_FILE_NAMES



fetchThreads inputBuffers fillerThreads outputBufferSets writerThreads outputFiles

Data Source





Parameter: SESSION_PARAMETERS

- Definition: Session parameters of Oracle sessions.
- Implementation: ALTER SESSION SET command is invoked internally with the value of SESSION_PARAMETERS.

If ALTER SESSION SET is invoked by the command line, it's set for the current command line session only. It's NOT propagated to unload sessions since unload sessions are created during unload, and released after unload completed.



Parameter: LICENSE_KEY

- Definition: License key.
- Implementation:

EVALUATION: Limits thread sets up to 2, and limits number of unloaded rows per thread set up to 10,000,000. Total unloaded row count limit is 20,000,000.

All other features are available.

- NULL: Generates pre-license key. ubTools provides production license key by pre-license key.
- <licenseKey>: Production license key provided by ubTools.

UNLOAD performance tips:

FetchThread

- If FillerThread wait time is high, tune FillerThread.
- If Database+Network+OCI+Other wait time is high, tune as below:
 - if it's supported, use `UB_PARALLEL` hint. If single or near single partition/subpartition will be accessed, use `PARTITION_SPLIT/SUBPARTITION_SPLIT` as an object split level.
 - Set `ARRAY_FETCH_ROW_COUNT` to an optimized value. The default value is usually optimized for many unloads.
 - If `OUTPUT_FILE_FORMAT=TEXT`, then set client `NLS_LANG` environment variable to database character set. This will eliminate character set conversion between client and server.
 - Tune SQL,SQL*Net,Network.

UNLOAD performance tips:

FillerThread

- If WriterThread wait time is high, tune as below:
 - Increase OUTPUT_BUFFERS_PER_FILE.
 - Tune WriterThread.
- Increase number of output files. This will increase number of FillerThread.

UNLOAD performance tips:

WriterThread

- If kernel CPU usage is high while copying output buffers to file system cache, then set `BYPASS_FILE_SYSTEM_CACHE=TRUE`.
- If kernel CPU usage of unload is not high, setting `BYPASS_FILE_SYSTEM_CACHE=TRUE` may not improve performance and it may degrade performance. In this case, set `BYPASS_FILE_SYSTEM_CACHE=FALSE`.
- Don't read output files during unload if `BYPASS_FILE_SYSTEM_CACHE=TRUE`. Unload may be switched to file system cache by kernel. This switch is not controlled by ubSQL and may not be optimized.
- If a process has to read output files during unload, set `BYPASS_FILE_SYSTEM_CACHE=FALSE`.
- Consider unloading to pipes instead of regular files.



ubTools

ubSQL 6.0