Oracle® SQLcl

User's Guide

Release 19.2
F20547-01
August 2019

Oracle SQLcl User's Guide, Release 19.2

F20547-01

Primary Author: Celin Cherian

Contributing Authors: Tulika Das

Contributors: Syme Kutz, Jeff D. Smith

Contents

Preface

This guide provides usage information about Oracle SQLcl (SQL Developer Command Line), a Java-based command-line interface for Oracle Database.

Audience

This guide is intended for those using Oracle SQLcl.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Working with Oracle SQLcl

Oracle SQLcl (SQL Developer Command Line) is a Java-based command-line interface for Oracle Database. Using SQLcl, you can interactively or batch execute SQL and PL/SQL statements. SQLcl provides inline editing, statement completion, command recall, and also supports existing SQL*Plus scripts.

This chapter contains the following topics:

1.1 Alphabetic List of SQLcl Commands

@{url | file_name[.ext]} [arg ...]

@@ { url | file_name[.ext] } [arg ...]

/ (slash)

ACC[EPT] *variable* [NUM[BER] | CHAR | DATE | BINARY_FLOAT | BINARY_DOUBLE] [FOR[MAT] *format*] [DEF[AULT] *default*] [PROMPT *text* | NOPR[OMPT]] [HIDE]

ALIAS [<*name*>=<*SQL statement*>;| LOAD [<*filename*>]|SAVE [<*filename*>] | LIST [<*NAME*>] | DROP <*name*> | DESC <*name*> <*Description String*>]

APEX [export <application_id>]

A[PPEND] *text*

ARCHIVE LOG LIST

BRE[AK] [ON *report_element* [*action* [*action*]]] ...

BRIDGE

BTI[TLE] [*printspec* [*text* | *variable*] ...] | [ON | OFF]

CD [<directory>]

C[HANGE] *sepchar old* [*sepchar* [*new* [*sepchar*]]]

```
CL[EAR] option ...

COL[UMN] [{column | expr} [option ...]]

COMP[UTE] [function [LAB[EL] text] ... OF {expr | column | alias} ...ON
{expr | column | alias | REPORT | ROW} ...]

CONN[ECT] [{<logon>| / |proxy} [AS {SYSOPER | SYSDBA | SYSASM}] [edi-
tion=value]]

CTAS table new_table

COPY {FROM database | TO database | FROM database TO database} {APPEND |
CREATE | INSERT | REPLACE | APPEND_BYTE | CREATE_BYTE | REPLACE_BYTE} des-
tination_table[(column, column, column, ...)] USING query

DDL [object_name [type] [SAVE filename]]

DEF[INE] [variable] | [variable = text]

DEL [n | n m | n * | n LAST | * | * n | * LAST | LAST]

DESC[RIBE] {[schema.]object[@connect_identifier]}

DISC[ONNECT]

ED[IT] [file_name[.ext]]

EXEC[UTE] statement

{EXIT | QUIT} [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable]
[COMMIT | ROLLBACK]

FIND [<filename>]

FORMAT [BUFFER | RULES <filename> | FILE <input_file> <output_file>]

GET [FILE] file_name[.ext] [LIST | NOLIST]

HELP | ? [topic]

HISTORY [index | FULL | USAGE | SCRIPT | TIME | CLEAR (SESSION)?]

HO[ST] [command]

INFO[RMATION] {[schema.]object[@connect_identifier]}

I[NPUT] [text]

L[IST] [n | n m | n * | n LAST | * | * n | * LAST | LAST]

LOAD [schema.]table_name[@db_link] file_name

OERR <facility> <error>

PASSW[ORD] [username]

PAU[SE] [text]

PRINT [variable ...]

PRO[MPT] [text]
```

```
{QUIT | EXIT} [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable]
[COMMIT | ROLLBACK]

REM[ARK]

REPEAT <iterations> <sleep>

REST [export [<module_name> | <module_prefix>] | modules | privileges |
schemas]

R[UN]

SAV[E] [FILE] file_name[.ext] [CRE[ATE] | REP[LACE] | APP[END]]

SCRIPT <script file>

SET system_variable value

SHO[W] [option]

SHUTDOWN [ABORT | IMMEDIATE | NORMAL | TRANSACTIONAL [LOCAL]]

SODA

SPO[OL] [filename[.ext] [CRE[ATE] | REP[LACE] | APP[END]] | OFF | OUT]

SSHTUNNEL <username>@<hostname> -i <identity_file> [-L localPort:Remote-
host:RemotePort]

STA[RT] { url | file_name[.ext] } [arg ...]

STARTUP db_options | cdb_options | upgrade_options

STORE {SET} file_name[.ext] [CRE[ATE] | REP[LACE] | APP[END]]

TNSPING <address>

TTI[TLE] [printspec [text | variable] ...] | [ON | OFF]

UNDEF[INE] variable ...

WHENEVER OSERROR {EXIT [SUCCESS | FAILURE | n | variable | :BindVariable]
[COMMIT | ROLLBACK] | CONTINUE[COMMIT | ROLLBACK | NONE]}

WHENEVER SQLERROR {EXIT [SUCCESS | FAILURE | WARNING | n | variable
| :BindVariable] [COMMIT | ROLLBACK] | CONTINUE [COMMIT | ROLLBACK |
NONE]}

WHICH <filename>

XQUERY xquery_statement
```

> **Note:**
>
> You can use the up and down arrow keys to cycle through the previous 100 statements or scripts.

1.2 List of Unsupported Commands and Features in SQL*Plus

Commands

- `REPHEADER`
- `REPFOOTER`
- `TIMING`

The `TIMING` command is replaced by the `SET TIMING` command.

System Variables and Environment Settings through the SET Command

- `autoprint`
- `autorecovery`
- `cmdsep`
- `copytypecheck`
- `describe`
- `eschar`
- `flagger`
- `flush`
- `fullcolname`
- `logsource`
- `loboffset`
- `markup`
- `recsep`
- `recsepchar`
- `shiftinout`
- `sqlcase`
- `sqlprefix`
- `sqlterminator`
- `tab`
- `underline`
- `xmloptimizationcheck`

1.3 Starting and Leaving SQLcl

Use the following commands to log in to and out of SQLcl.

`SQLCL [[`*`option`*`] [`*`logon`*` | / NOLOG] [`*`start`*`]]`

where *option* has the following syntax:

```
-H[ELP] | -V[ERSION] | [ [-C[OMPATIBILITY] x.y[.z]]] [-L[OGON]] [-NOLOGIN-
TIME] [-R[ESTRICT] {1 | 2 | 3}] [-S[ILENT]] [-AC]]
```

where *logon* has the following syntax:

```
{username[/password] [@connect_identifier] | /} [AS {SYSASM |SYSBACKUP |
SYSDBA |SYSDG |SYSOPER |SYSRAC |SYSKM}] [edition=value]
```

and where *start* has the following syntax:

```
@{url | file_name[.ext]} [arg ...]
```

```
{EXIT | QUIT} [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable]
[COMMIT | ROLLBACK]
```

Commits or rolls back all pending changes, logs out of Oracle, terminates SQLcl and returns control to the operating system.

```
{QUIT | EXIT} [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable]
[COMMIT | ROLLBACK]
```

Commits or rolls back all pending changes, logs out of Oracle, terminates SQLcl and returns control to the operating system.

1.4 Starting Up and Shutting Down a Database

Starting up and shutting down a database requires DBA privileges.

```
STARTUP db_options | cdb_options | upgrade_options
```

where *db options* has the following syntax:

```
[FORCE] [RESTRICT] [PFILE=filename] [QUIET] [ MOUNT [dbname] | [ OPEN
[open_db_options] [dbname] ] | NOMOUNT ]
```

where *open_db_options* has the following syntax:

```
READ {ONLY | WRITE [RECOVER]} | RECOVER
```

where *cdb_options* has the following syntax:

```
root_connection_options | pdb_connection_options
```

where *root_connection_options* has the following syntax:

```
PLUGGABLE DATABASE pdbname [FORCE] | [RESTRICT] [ OPEN {open_pdb_options}]
```

where *pdb_connection_options* has the following syntax:

```
[FORCE] | [RESTRICT] [ OPEN {open_pdb_options}]
```

where *open_pdb_options* has the following syntax:

```
READ WRITE | READ ONLY
```

and where *upgrade_options* has the following syntax:

```
[PFILE=filename] {UPGRADE | DOWNGRADE} [QUIET]
```

Starts an Oracle Database instance with several options, including mounting and opening a database.

`SHUTDOWN [ABORT | IMMEDIATE | NORMAL | TRANSACTIONAL [LOCAL]]`

Shuts down a currently running Oracle instance, optionally closing and dismounting a database.

1.5 Entering and Executing Commands

Use the following commands to execute and collect timing statistics on SQL commands and PL/SQL blocks:

`/ (slash)`

Executes the most recently executed SQL command or PL/SQL block which is stored in the SQL buffer. Does not list the command. Use slash (/) at the command prompt or line number prompt in SQLcl command line.

`EXEC[UTE] statement`

Executes a single PL/SQL statement or runs a stored procedure.

`R[UN]`

Lists and executes the most recently executed SQLcl command or PL/SQL block which is stored in the SQL buffer. The buffer has no command history list and does not record SQLcl commands.

`TIMI[NG]`

Timing is only available as a switch.

Use the following command to access the help system:

`HELP | ? [topic]`

Accesses the command-line help system. Enter `HELP INDEX` or `? INDEX` for a list of topics. You can view the Oracle Database Library at http://www.oracle.com/technology/documentation.

Use the following command to execute operating system commands:

`HO[ST] [command]`

Executes an operating system command without leaving SQLcl. Enter `HOST` without command to display an operating system prompt. You can then enter multiple operating system commands.

With some operating systems, you can use another character instead of HOST such as "!" (UNIX) and "$" (Windows). See the Oracle installation and user's manuals provided for your operating system for details.

Use the following command to recall the history of SQLcl commands:

`HISTORY [index | FULL | USAGE | SCRIPT | TIME | CLEAR (SESSION)?] | FAILS`

- Use the Up and Down arrow keys to navigate through history items at the prompt.
- Use the `HISTORY` command to print the history contents.
- History is limited to the last 100 statements.

- `SET HISTORY LIMIT N` allows you to change the default limit, where *N* is the maximum number.
- History is retained between SQLcl sessions.
- By default, the `SHOW`, `HISTORY`, `CONNECT`, and `SET` commands are not saved in history.
- `SET HISTORY BLACKLIST` allows you to set the commands that should not be recorded in history.

1.6 Manipulating SQL, SQLcl, and PL/SQL Commands

Use the following commands to edit SQL commands and PL/SQL blocks:

`A[PPEND] text`

Adds specified text to the end of the current line in the SQL buffer. To separate *text* from the preceding characters with a space, enter two spaces. To append *text* that ends with a semicolon, end the command with two semicolons (a single semicolon is interpreted as a command terminator).

`C[HANGE] sepchar old [sepchar [new [sepchar]]]`

Changes first occurrence of *old* on the current line of the SQL buffer. The buffer has no command history list and does not record SQLcl commands. You can use any non-alphanumeric character such as "/" or "!" as a *sepchar*. You can omit the space between `CHANGE` and the first *sepchar*.

`DEL [n | n m | n * | n LAST | * | * n | * LAST | LAST]`

Deletes one or more lines of the SQL buffer ("*" indicates the current line). You can omit the space between `DEL` and *n* or *, but not between `DEL` and `LAST`. Enter `DEL` with no clauses to delete the current line of the buffer. The buffer has no command history list and does not record SQLcl commands.

`I[NPUT] [text]`

Adds one or more new lines of text after the current line in the SQL buffer. The buffer has no command history list and does not record SQLcl commands.

`L[IST] [n | n m | n * | n LAST | * | * n | * LAST | LAST]`

Lists one or more lines of the most recently executed SQL command or PL/SQL block which is stored in the SQL buffer. Asterisk (*) indicates the current line. You can omit the space between `LIST` and *n* or *, but not between `LIST` and `LAST`. Enter `LIST` with no clauses to list all lines.

In SQLcl, you can also use ";" to list all the lines in the SQL buffer. The buffer has no command history list and does not record SQLcl commands.

Use the following commands to run scripts:

`@ { url | file_name[.ext] } [arg ...]`

Runs the SQLcl statements in the specified script. The script can be called from the local file system or a web server. You can pass values to script variables in the usual way.

`@@ { url | file_name[.ext] } [arg ...]`

Runs the SQLcl statements in the specified script. This command is almost identical to the @ command. It is useful for running nested scripts because it has the additional functionality of looking for the specified script in the same path or *url* as the calling script.

`REPEAT <iterations> <sleep>`

Repeats the current SQL in the buffer at the specified times with sleep intervals. The maximum sleep interval is 120 seconds.

`SCRIPT <script file>`

Runs the SQLcl statements in the specified script.

`STA[RT] { url | file_name[.ext] } [arg ...]`

Runs the SQLcl statements in the specified script. The script can be called from the local file system or a web server. You can pass values to script variables in the usual way.

Use the following commands to create and modify scripts:

`ED[IT] [file_name[.ext]]`

Invokes an operating system text editor on the contents of the specified file or on the contents of the SQL buffer. To edit the buffer contents, omit the file name.

The DEFINE variable _EDITOR can be used to set the editor to use. In SQLcl, _EDITOR can be set to any editor that you prefer. *Inline* will set the editor to be the SQLcl editor. This supports the following shortcuts:

- `^R` - Run the current buffer
- `^W` - Go to top of buffer
- `^S` - Go to bottom of buffer
- `^A` - Go to start of line
- `^E` - Go to end of line

`FORMAT`

- `FORMAT BUFFER` - formats the script in the SQLcl Buffer
- `FORMAT RULES <filename>` - Loads SQLDeveloper Formatter rules file to formatter
- `FORMAT FILE <input_file> <output_file>`

`GET file_name[.ext] [LIST | NOLIST]`

Loads a SQL statement or PL/SQL block from a file into the SQL buffer. The buffer has no command history list and does not record SQLcl commands.

`REM[ARK]`

Begins a comment in a script. The REMARK command must appear at the beginning of a line, and the comment ends at the end of the line (a line cannot contain both a comment and a command). SQLcl does not interpret the comment as a command.

`SAV[E] [FILE] file_name[.ext] [CRE[ATE] | REP[LACE] | APP[END]]`

Saves the contents of the SQL buffer in a script. The buffer has no command history list and does not record SQLcl commands.

`STORE {SET} file_name[.ext] [CRE[ATE] | REP[LACE] | APP[END]]`

Saves attributes of the current SQLcl environment in a file.

`WHENEVER OSERROR {EXIT [SUCCESS | FAILURE | n | variable | :BindVariable] [COMMIT | ROLLBACK] | CONTINUE [COMMIT | ROLLBACK | NONE]}`

Performs the specified action (exits SQLcl by default) if an operating system error occurs (such as a file writing error).

`WHENEVER SQLERROR {EXIT [SUCCESS | FAILURE | WARNING | n | variable | :BindVariable] [COMMIT | ROLLBACK] | CONTINUE [COMMIT | ROLLBACK | NONE]}`

Performs the specified action (exits SQLcl by default) if a SQL command or PL/SQL block generates an error.

Use the following commands to write interactive commands:

`ACC[EPT] variable [NUM[BER] | CHAR | DATE | BINARY_FLOAT | BINARY_DOUBLE] [FOR[MAT] format] [DEF[AULT] default] [PROMPT text | NOPR[OMPT]] [HIDE]`

Reads a line of input and stores it in a given substitution variable.

`DEF[INE] [variable] | [variable = text]`

Specifies a substitution variable and assigns a CHAR value to it, or lists the value and variable type of a single variable or all variables.

`PAU[SE] [text]`

Displays the specified text then waits for the user to press RETURN.

`PRO[MPT] [text]`

Sends the specified message or a blank line to the user's screen.

`UNDEF[INE] variable ...`

Deletes one or more substitution variables that you defined either explicitly (with the DEFINE command) or implicitly (with a START command argument).

Use the following commands to create and display bind variables:

`PRINT [variable ...]`

Displays the current values of bind variables, or lists all bind variables.

Use the following symbols to create substitution variables and parameters for use in scripts:

&n

Specifies a parameter in a script you run using the START command. START substitutes values you list after the script name as follows: the first for &1, the second for &2, and so on.

&user_variable, &&user_variable

Indicates a substitution variable in a SQL or SQLcl command. SQLcl substitutes the value of the specified substitution variable for each substitution variable it encounters. If the substitution variable is undefined, SQLcl prompts you for a value *each* time an "&" variable is found, and the *first* time an "&&" variable is found.

`. (period)`

Terminates a substitution variable followed by a character that would otherwise be part of the variable name.

1.7 Formatting Query Results

Use the following commands to format, store and print your query results.

`BRE[AK] [ON report_element [action [action]]] ...`

Specifies where changes occur in a report and the formatting action to perform, such as:

- suppressing the display of duplicate values for a given column
- skipping a line each time a given column value changes
- printing computed figures each time a given column value changes or at the end of the report

Enter `BREAK` with no clauses to list the current BREAK definition.

Where *report_element* has the following syntax:

`{column | expression | ROW | REPORT}`

and where *action* has the following syntax:

`[SKI[P] n | [SKI[P]] PAGE] [NODUP[LICATES] | DUP[LICATES]]`

`BTI[TLE] [printspec [text | variable] ...] | [ON | OFF]`

Places and formats a title at the bottom of each report page, or lists the current BTI-TLE definition. Use one of the following clauses in place of *printspec*:

```
BOLD
CE[NTER]
COL n
FORMAT text
LE[FT]
R[IGHT]
S[KIP] [n]
TAB n
```

`CL[EAR] option ...`

Resets or erases the current value or setting for the specified option.

Where *option* represents one of the following clauses:

```
BRE[AKS]
BUFF[ER]
COL[UMNS]
COMP[UTES]
CONTEXT
SCR[EEN]
SQL
```

```
TIMI[NG]
```

`COL[UMN] [{column | expr} [option ...]]`

Specifies display attributes for a given column, such as:

- text for the column heading
- alignment for the column heading
- format for NUMBER data
- wrapping of column data

Also lists the current display attributes for a single column or for all columns.

Where *option* represents one of the following clauses:

```
ALI[AS] alias
CLE[AR]
ENTMAP {ON | OFF}
FOR[MAT] format
HEA[DING] text
JUS[TIFY] {L[EFT] | C[ENTER] | R[IGHT]}
LIKE {expr | alias}
NEWL[INE]
NEW_V[ALUE] variable
NOPRI[NT] | PRI[NT]
NUL[L] text
OLD_V[ALUE] variable
ON | OFF
WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]
```

> **Note:**
>
> Currently only `NEW_V[ALUE]` variable syntax is supported.

Enter `COLUMN [{column |expr} FORMAT format]` where the *format* element specifies the display format for the column.

To change the display format of a NUMBER column, use `FORMAT` followed by one of the elements in the following table:

Element	Examples	Description
, (comma)	`9,999`	Displays a comma in the specified position.
. (period)	`99.99`	Displays a period (decimal point) to separate the integral and fractional parts of a number.
$	`$9999`	Displays a leading dollar sign.

Element	Examples	Description
0	0999 9990	Displays leading or trailing zeros (0).
9	9999	Displays a value with the number of digits specified by the number of 9s. Value has a leading space if positive, a leading minus sign if negative. Blanks are displayed for leading zeros. A zero (0) is displayed for a value of zero.
B	B9999	Displays blanks for the integer part of a fixed-point number when the integer part is zero, regardless of zeros in the format model.
C	C999	Displays the ISO currency symbol in the specified position.
D	99D99	Displays the decimal character to separate the integral and fractional parts of a number.
EEEE	9.999EEEE	Displays a value in scientific notation (format must contain exactly four "E"s).
G	9G999	Displays the group separator in the specified positions in the integral part of a number.
L	L999	Displays the local currency symbol in the specified position.
MI	9999MI	Displays a trailing minus sign after a negative value. Displays a trailing space after a positive value.
PR	9999PR	Displays a negative value in <angle brackets>. Displays a positive value with a leading and trailing space.
RN rn	RN rn	Displays uppercase Roman numerals. Displays lowercase Roman numerals. Value can be an integer between 1 and 3999.
S	S9999 9999S	Displays a leading minus or plus sign. Displays a trailing minus or plus sign.
TM	TM	Displays the smallest number of decimal characters possible. The default is TM9. Fixed notation is used for output up to 64 characters, scientific notation for more than 64 characters. Cannot precede TM with any other element. TM can only be followed by a single 9 or E.
U	U9999	Displays the dual currency symbol in the specified position.

COMP[UTE] [*function* [LAB[EL] *text*] ... OF {*expr* | *column* | *alias*} ...ON {*expr* | *column* | *alias* | REPORT | ROW} ...]

In combination with the BREAK command, calculates and prints summary lines using various standard computations. It also lists all COMPUTE definitions. The following table lists valid functions. All functions except NUMBER apply to non-null values only. COMPUTE functions are always executed in the following sequence AVG, COUNT, MINIMUM, MAXIMUM, NUMBER, SUM, STD, VARIANCE.

Function	Computes	Applies to Datatypes
AVG	Average of non-null values	NUMBER
COU[NT]	Count of non-null values	All types
MIN[IMUM]	Minimum value	NUMBER, CHAR, NCHAR, VARCHAR2 (VARCHAR), NVARCHAR2 (NCHAR VARYING)
MAX[IMUM]	Maximum value	NUMBER, CHAR, NCHAR, VARCHAR2 (VARCHAR), NVARCHAR2 (NCHAR VARYING)
NUM[BER]	Count of rows	All types
SUM	Sum of non-null values	NUMBER
STD	Standard deviation of non-null values	NUMBER
VAR[IANCE]	Variance of non-null values	NUMBER

`SET SQLFORMAT {csv | html | xml | json | ansiconsole | insert | loader | fixed | default}`

Outputs reports in various formats. The `ansiconsole` option formats and resizes data according to the column widths, for easier readability. The `json` option returns a query in JSON format.

`SET SQLFORMAT DELIMITED <delimiter> <left enclosure> <right enclosure>` allows you to set a custom delimited format.

`SET SQLFORMAT JSON-FORMATTED` returns a query in well formatted JSON output.

`SPO[OL] [filename[.ext] [CRE[ATE] | REP[LACE] | APP[END]] | OFF | OUT]`

Stores query results in a file, or optionally sends the file to a printer. `OFF` stops spooling. `OUT` stops spooling and sends the file to your computer's default printer. Enter `SPOOL` with no clauses to list the current spooling status. If no file extension is given, the default extension, *`.lst`* or *`.lis`*, is used.

`TTI[TLE] [printspec [text | variable] ...] | [ON | OFF]`

Places and formats a specified title at the top of each report page, or lists the current TTITLE definition. The old form of TTITLE is used if only a single word or a string in quotes follows the TTITLE command.

Where *`printspec`* represents one or more of the following clauses:

```
BOLD
CE[NTER]
COL n
FORMAT text
LE[FT]
R[IGHT]
S[KIP] [n]
TAB n
```

1.8 Accessing Databases

Use the following commands to access and copy data between tables on different databases:

```
CONN[ECT] [{<logon>| / |proxy} [AS {SYSOPER | SYSDBA | SYSASM}] [edition=value]]
```

where *logon* has the following syntax:

```
username[/password] [@connect_identifier]
```

where *proxy* has the following syntax:

```
proxyuser[username] [/password] [@connect_identifier]
```

> **Note:**
>
> The brackets around *username* in *proxy* are required syntax.

Connects a given username to the Oracle Database. If you omit *connect_identifier*, SQLcl connects you to the default database. If you omit *username* and/or *password*, SQLcl prompts you for them. CONNECT followed by a slash (/) connects you using a default (OPS$) logon.

When you run a CONNECT command, the site profile, glogin.sql, and the user profile, login.sql, are processed in that order. CONNECT does not reprompt for username or password if the initial connection does not succeed.

```
DISC[ONNECT]
```

Commits pending changes to the database and logs the current user out of Oracle, but does not exit SQLcl. In SQLcl command line, use EXIT or QUIT to log out of Oracle and return control to your computer's operating system.

```
COPY {FROM database | TO database | FROM database TO database} {APPEND | CREATE | INSERT | REPLACE | APPEND_BYTE | CREATE_BYTE | REPLACE_BYTE} destination_table[(column, column, column, ...)] USING query
```

where *database* has the following syntax:

```
username[/password]@connect_identifier
```

Copies data from a query to a table in the same or another database. APPEND, CREATE, INSERT or REPLACE specifies how COPY treats the existing copy of the destination table (if it exists). USING *query* identifies the source table and determines which rows and columns COPY copies from it. COPY supports CHAR, DATE, LONG, NUMBER and VARCHAR2 datatypes.

```
PASSW[ORD] [username]
```

Allows you to change a password without displaying it on an input device.

```
XQUERY xquery_statement
```

Allows you to run an XQuery from SQLcl.

1.9 Miscellaneous Commands

`ALIAS [<name>=<SQL statement>;| LOAD [<filename>]|SAVE [<filename>] | LIST [<NAME>] | DROP <name> | DESC <name> <Description String>]`

Alias is a command which allows you to save a SQL, PL/SQL or SQL*Plus script and assign it a shortcut command.

- `ALIAS` — Print a list of aliases
- `ALIAS LIST <alias_name>` — List the contents of the alias

The following example shows how to create a simple alias:

```
SQL> ALIAS action1=select :one from dual;
```

> **Note:**
>
> Define an alias simply by using the alias keyword followed by a single identifier name followed by an '='. Anything after the '=' will be used as the alias contents. If it is SQL, it will be terminated by ';'. If it is PL/SQL, it will be terminated by '/'.

`APEX`

Lists Application Express Applications. Use `APEX EXPORT <app id>` to export the application which could be combined with spool for writing to a file.

`ARCHIVE LOG LIST`

Displays information about redo log files.

`BRIDGE <targetTableName> as "<jdbcURL>"(<sqlQuery>);`

Used mainly to script data move between two connections/schemas. It also includes functionality to dynamically create Oracle tables which "fit" the data being received through JDBC. The following functionality is available:

1. Query tables in other connections
2. Query tables in multiple connections in the same statement
3. Insert data from one connection into another
4. Create a table and insert data into it from another connection

`CTAS table new_table`

Uses DBMS_METADATA to extract the DDL for the existing table, then modifies that into a create table as select * from.

`DDL [object_name [type] [SAVE filename]]`

Generates the code to reconstruct the object listed. Use the *type* option for materialized views. Use the SAVE option to save the DDL to a file.

```
DESC[RIBE] {[schema.]object[@connect_identifier]}
```

Lists the column definitions for a table, view or synonym, or the specifications for a function or procedure.

```
FIND [<filename>]
```

Searches the SQLPATH and its directories for the specified file name. `FIND where <filename>` lists all the SQLPATH locations where it finds files matching the specified file name.

```
INFO[RMATION] {[schema.]object[@connect_identifier]}
```

Lists more detailed information about the column definitions for a table, view or synonym, or the specifications for a function or procedure.

> **Note:**
>
> INFORMATION+ will show column statistics.

```
LOAD [schema.]table_name[@db_link] file_name
```

Loads a comma separated value (csv) file into a table. The first row of the file must be a header row. The columns in the header row must match the columns defined on the table. The columns must be delimited by a comma and may optionally be enclosed in double quotes. Lines can be terminated with standard line terminators for Windows, UNIX, or Mac. File must be encoded UTF8.

The load is processed with 50 rows per batch. If AUTOCOMMIT is set in SQLcl, a commit is done every 10 batches. The load is terminated if more than 50 errors are found.

```
OERR <facility> <error>
```

Displays information about errors. Facility is identified by the prefix string in the error message. For example, if you get ORA-7300, "ora" is the facility and "7300" is the error. So you should type "oerr ora 7300".

```
REST
```

REST allows you to export Oracle REST Data Services 3.x services. This is applicable for Oracle REST Data Services release 3.0.5 or later. If you have an earlier version of Oracle REST Data Services, you will need to upgrade. See the Installing Oracle REST Data Services section in *Oracle REST Data Services Installation, Configuration, and Development Guide* for details.

The options are:

- `REST export` — Export all Oracle REST Data Services 3.x service modules
- `REST export <module_name>` — Export a specific module
- `REST export <module_uri_uri prefix>` — Export a specific module related to the given prefix
- `REST modules` — List the available modules
- `REST privileges` — List the existing privileges

- `REST schemas` — List the available schemas

`SODA`

SODA allows schemaless application development using the JSON data model. The options are:

- `SODA create <collection_name>` — Create a new collection
- `SODA list` — List all the collections
- `SODA get <collection_name> [-all | -f | -k | -klist] [{<key> | <k1> <k2> ... | <qbe>}]` — List documents the collection. Optional arguments:
  - `all` : list the keys of all docs in the collection
  - `k` : list docs matching the specific <*key*>
  - `klist` : list docs matching the list of keys
  - `f` : list docs matching the <*qbe*>
- `SODA insert <collection_name> <json_str | filename>` — Insert a new document within a collection
- `SODA drop <collection_name>` — Delete existing collection
- `SODA count <collection_name> [<qbe>]` — Count number of documents inside collection. Optional parameter <*qbe*> returns number of matching documents
- `SODA replace <collection_name> <oldkey> <new_{str | doc}>` — Replace one document with another
- `SODA remove <collection_name> [-k | -klist | -f] {<key> | <k1> <k2> ...| <qbe>}` — Remove documents from collection. Optional arguments:
  - `k` : Remove document in collection matching the specific <*key*>
  - `klist` : Remove document in collection matching the list <*key1*> <*key2*> ...
  - `f` : Remove document in collection matching <*qbe*>

`SET system_variable value`

Sets a system variable to alter the SQLcl environment settings for your current session. For example, to:

- Set the display width for data
- Customize HTML formatting
- Enable or disable printing of column headings
- Set the number of lines per page

Enter a system variable followed by a value as shown below:

```
SET APPI[NFO]{ON | OFF | text}
SET ARRAY[SIZE] {15 | n}
SET AUTO[COMMIT] {ON | OFF | IMM[EDIATE] | n}
SET AUTOP[RINT] {ON | OFF}
SET AUTORECOVERY {ON | OFF]
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]}
SET BLO[CKTERMINATOR] {. | c | ON | OFF}
SET CLEAR [ TOP | BOTTOM | SAME ]
```

```
SET CMDS[EP] {; | c | ON | OFF}
SET COLSEP {_ | text}
SET CON[CAT] {. | c | ON | OFF}
SET COPYC[OMMIT] {0 | n}
SET COPYTYPECHECK {ON | OFF}
SET DDL [[ PRETTY | SQLTERMINATOR | CONSTRAINTS | REF_CONSTRAINTS |
CONSTRAINTS_AS_ALTER|OID | SIZE_BYTE_KEYWORD | PARTITIONING | SEG-
MENT_ATTRIBUTES | STORAGE | TABLESPACE | SPECIFICATION | BODY | FORCE |
INSERT | |INHERIT | RESET] {on|off} ] | OFF ]
SET DEF[INE] {& | c | ON | OFF}
SET ECHO {ON | OFF}
SET EDITF[ILE] file_name[.ext]
SET EMB[EDDED] {ON | OFF}
SET ENCODING
SET ERRORL[OGGING] {ON | OFF} [TABLE [schema.]tablename] [TRUNCATE]
[IDENTIFIER identifier]
SET ESC[APE] {\ | c | ON | OFF}
SET ESCCHAR {@ | ? | % | $ | OFF}
SET EXITC[OMMIT] {ON | OFF}
SET [EXP[LAIN]] [STAT[ISTICS]]
SET FEED[BACK] {6 | n | ON | OFF}
SET FLU[SH] {ON | OFF}
SET HEA[DING] {ON | OFF}
SET HEADS[EP] {| | c | ON | OFF}
SET INSTANCE [instance_path | LOCAL]
SET LDAPCON
SET LIN[ESIZE] {80 | n}
SET LOBOF[FSET] {n | 1}
SET LOGSOURCE [pathname]
SET LONG {80 | n}
SET LONGC[HUNKSIZE] {80 | n}
SET NET {ON | OFF | READONLY}
SET NEWP[AGE] {1 | n | NONE}
SET NOVERWRITE {ON | OFF | WARN}
SET NULL text
SET NUMF[ORMAT] format
SET NUM[WIDTH] {10 | n}
SET PAGES[IZE] {14 | n}
SET PAU[SE] {ON | OFF | text}
SET RECSEP {WR[APPED] | EA[CH] | OFF}
SET RECSEPCHAR {_| c}
SET SERVEROUT[PUT] {ON | OFF} [SIZE {n | UNL[IMITED]}] [FOR[MAT]
{WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]}]
SET SHIFT[INOUT] {VIS[IBLE] | INV[ISIBLE]}
SET SHOW[MODE] {ON | OFF}
SET SQLBL[ANKLINES] {ON | OFF}
SET SQLC[ASE] {MIX[ED] | LO[WER] | UP[PER]}
SET SQLCO[NTINUE] {> | text}
```

```
SET SQLFORMAT {csv | html | xml | json | ansiconsole | insert | loader
| fixed | default}
SET SQLN[UMBER] {ON | OFF}
SET SQLPLUSCOMPAT[IBILITY] {x.y[.z]}
SET SQLPRE[FIX] {# | c}
SET SQLP[ROMPT] {SQL> | text}
SET SQLT[ERMINATOR] {; | c | ON | OFF}
SET SUF[FIX] {SQL | text}
SET TAB {ON | OFF}
SET TERM[OUT] {ON | OFF}
SET TI[ME] {ON | OFF}
SET TIMI[NG] {ON | OFF}
SET TRIM[OUT] {ON | OFF}
SET TRIMS[POOL] {ON | OFF}
SET UND[ERLINE] {- | c | ON | OFF}
SET VER[IFY] {ON | OFF}
SET WRA[P] {ON | OFF}
```

```
SET DDL [[ PRETTY | SQLTERMINATOR | CONSTRAINTS | REF_CONSTRAINTS | CON-
STRAINTS_AS_ALTER|OID | SIZE_BYTE_KEYWORD | PARTITIONING | SEGMENT_ATTRIB-
UTES | STORAGE | TABLESPACE | SPECIFICATION | BODY | FORCE | INSERT | |IN-
HERIT | RESET] {on|off} ] | OFF ]
```

Allows you to set the DDL transform option on DBMS_METADATA.

```
SET ENCODING <encoding>
```

Allows you to set the encoding for the current session. Use `SHOW ENCODING` to view the encoding set for the current session. Use SHOW ENCODINGS to list the encodings available on your platform.

```
SHO[W] [option]
```

Shows the value of a SQLcl system variable, or the current SQLcl environment. Enter any system variable set by the SET command in place of *system_variable*. SHOW SGA can only be used by a DBA user. Use one of the following terms or clauses in place of *option*:

```
system_variable
ALL
BTI[TLE]
CON_ID
CON_NAME
CONNECTION
DDL
EDITION
ENCODING
ENCODINGS
ERR[ORS] [ {FUNCTION | PROCEDURE | PACKAGE | PACKAGE BODY | TRIGGER |
VIEW | TYPE | TYPE BODY | DIMENSION | JAVA CLASS} [schema.]name]
INSTANCE
JAVA
JDBC
```

```
LNO
NLS
PARAMETER[S] [parameter_name]
PDBS
PNO
RECYC[LEBIN] [original_name]
REL[EASE]
REPF[OOTER]
REPH[EADER]
SGA
SPOO[L]
SPPARAMETER[S] [parameter_name]
SQLCODE
SQLPATH
TNS
TTI[TLE]
USER
VERSION
```

`SHOW ENCODING`

Shows the encoding which is set for the client.

`SHOW ENCODINGS`

Shows the available encodings for the client.

`SSHTUNNEL <username>@<hostname> -i <identity_file> [-L localPort:Remote-host:RemotePort]`

Creates a tunnel using standard ssh options such as port forwarding like option -L of the given port on the local host will be forwarded to the given remote host and port on the remote side. It also supports identity files, using the ssh -i option. If passwords are required, they will be prompted for.

`TNSPING <address>`

The TNSPING utility determines whether the listener for a service on an Oracle Net network can be reached successfully.

`WHICH`

Searches the SQLPATH and its directories for the specified file name and prints the name of the first file matching the specified file name in the SQLPATH.

2
Using Liquibase with SQLcl

This chapter explains the Liquibase feature in SQLcl. It has the following topics:

2.1 About Liquibase

Liquibase is an open-source database-independent library for tracking, managing and applying database schema changes. For an understanding of the major concepts in Liquibase, see Major Concepts.

In SQLcl, you can now execute commands to generate a changelog for a single object or for a full schema (changeset and changelogs). You can process these objects manually using SQLcl or through any of the traditional Liquibase interfaces.

With the Liquibase feature in SQLcl, you can:

- Generate and execute single object changelogs
- Generate and execute schema changesets with object dependencies
- Automatically sort a changeset during creation based on object dependencies
- Record all SQL statements for changeset or changelog execution, as it is generated
- Provide full rollback support for changesets and changelogs automatically

2.2 Requirements for Using Liquibase

The two important aspects for using Liquibase are capturing and deploying objects in an Oracle database.

Capture Objects

To capture an object or a schema, you must have SQLcl 19.2 or later installed.

In this release, you can only capture objects from the schema you are connected to in SQLcl. You also need write permission on the directory in which you save the files.

If you are capturing an entire schema, the user you are connected to must have the privilege to create a table. The DATABASECHANGELOG_EXPORT table is created internally to gather object details and sort them correctly. The created object is automatically excluded from the capture process and destroyed upon capture completion.

Deploy Objects

Liquibase uses the DATABASECHANGELOG table to track the changesets that have been run. The DATABASECHANGELOGLOCK table ensures that only one instance of Liquibase is running at a time. The DATABASECHANGELOG_EXPORT table tracks the object state and the SQL statements executed during deployment.

- **SQLcl**

  Deploying changes to any database through SQLcl requires the 19.2 release or later and the privilege to create a table. You must have necessary permissions to create any object type through the change that you are deploying.

- **Liquibase**

  If you use Liquibase directly to deploy changesets, you need to have:

  – the extension installed in your Liquibase environment. Add the `oracle-liquibase.jar` file in `liquibase/lib/ext`.

  – the privilege to create a table.

2.3 Supported Types

DDL types use create or replace syntax, however, a snapshot of the object is taken before applying the change so automatic rollback to the last known state is supported. SXML types support automatic alter generation with automatic rollback support.

DDL types have their own change type.

- CONSTRAINT
- DIMENSION
- FUNCTION
- OBJECT_GRANT
- PACKAGE_BODY
- PACKAGE_SPEC
- PROCEDURE
- PUBLIC_SYNONYM
- REF_CONSTRAINT
- SYNONYM
- TYPE BODY
- TYPE SPEC

SXML types share the SXML change type.

- AQ_QUEUE
- AQ_QUEUE_TABLE
- AQ_TRANSFORM MATERIALIZED_VIEW
- ASSOCIATION
- AUDIT

- AUDIT_OBJ
- CLUSTER
- CONTEXT
- DB_LINK
- DEFAULT_ROLE
- FGA_POLICY
- JOB
- LIBRARY
- MATERIALIZED_VIEW_LOG
- OPERATOR
- PROFILE
- PROXY
- REFRESH_GROUP
- RESOURCE_COST
- RLS_CONTEXT
- RLS_GROUP
- RMGR_CONSUMER_GROUP
- RMGR_INTITIAL_CONSUMER_GROUP
- RMGR_PLAN
- RMGR_PLAN_DIRECTIVE
- ROLE
- ROLLBACK_SEGMENT
- SEQUENCE
- TABLE
- TABLESPACE
- TRIGGER XS_ACL
- TRUSTED_DB_LINK
- USER
- VIEW
- XMLSCHEMA
- XS_ACL_PARAM INDEX
- XS_DATA_SECURITY
- XS_ROLE
- XS_ROLESET
- XS_ROLE_GRANT
- XS_SECURITY_CLASS
- XS_USER

2.4 Supported Liquibase Commands in SQLcl

You can invoke the Liquibase commands in SQLcl with `liquibase` or `lb`. To display a list of all available commands, execute `liquibase` or `lb` with no arguments.

The different commands are:

`liquibase(lb) genobject <object_type> <object_name>`

Captures a single object using the current connection in SQLcl. It creates an xml file in the current working directory named <object name>_<object type>.xml.

Available Parameters:

Name	Description	Required
Object type	Type of object (this is an Oracle type)	Yes
Object name	Name of object	Yes

Example:

```
SQL> lb genobject table employees
lb genobject table employees
Action successfully completed please review created file EMPLOYEES_TA-
BLE.xml
```

`liquibase(lb) genschema`

Captures the entire schema that the user is connected to in SQLcl. It creates an xml file in the current working directory for each object in the schema, and a controller.xml file. The controller file is a change log that includes all files in the proper order to allow the schema to be deployed correctly.

Available Parameters:

Name	Description	Required
Public synonym	Capture public synonyms. Default is false.	No
Grants	Capture grants. Default is false.	No

Example:

```
SQL> lb genschema
lb genschema
[Type - TYPE_SPEC]:                          153 ms
[Type - TYPE_BODY]:                          29 ms
[Type - SEQUENCE]:                           48 ms
[Type - CLUSTER]:                            27 ms
[Type - TABLE]:                              36 ms
[Type - MATERIALIZED_VIEW_LOG]:              19 ms
[Type - MATERIALIZED_VIEW]:                  6 ms
```

```
[Type - VIEW]:                                148 ms
[Type - REF_CONSTRAINT]:                      272 ms
[Type - DIMENSION]:                           23 ms
[Type - FUNCTION]:                            27 ms
[Type - PROCEDURE]:                           64 ms
[Type - PACKAGE_SPEC]:                        171 ms
[Type - DB_LINK]:                             14 ms
[Type - SYNONYM]:                             22 ms
[Type - INDEX]:                               202 ms
[Type - TRIGGER]:                             51 ms
[Type - PACKAGE_BODY]:                        252 ms
[Method loadCaptureTable]:                    1864 ms
[Method parseCaptureTableRecords]:            7342 ms
[Method sortCaptureTable]:                    30 ms
[Method createExportChangeLogs]:              3 ms

Export Flags Used:
Export Grants               false
Export Synonyms             false
```

`liquibase(lb) gencontrolfile`

Creates an empty changelog, `master.xml`, with a placeholder to include files in the current working directory. You can use this command when you are creating you own changelog with custom changeset inclusions.

Example:

```
SQL> lb gencontrolfile
lb gencontrolfile
Action successfully completed please review created file controller.xml
```

`liquibase(lb) update <CHANGE LOG> {include schema}`

Applies the specified change log using the current connection. You can choose to have the schema included in the DDL by passing in `TRUE` for *include schema*. The default value is `FALSE`.

Example:

```
SQL> lb update foo_table.xml
lb update foo_table.xml

Table "FOO" created.
```

`liquibase(lb) updatesql <CHANGE LOG> {include schema}`

Generates and renders to the screen the SQL statements that would be applied for a specific change log. You can choose to have the schema included in the DDL by passing in `TRUE` for *include schema*. The default value is `FALSE`.

Example:

```
SQL> lb updatesql syme_table.xml
lb updatesql syme_table.xml
```

```
CREATE TABLE "Syme"
   (    "ID" NUMBER
   ) SEGMENT CREATION DEFERRED
  PCTFREE 10 PCTUSED 40 INITRANS 1 NOCOMPRESS LOGGING
  TABLESPACE "SYSAUX"
```

liquibase(lb) rollback <CHANGE LOG> <COUNT>

Rolls back changes starting from the last change applied using the input change log. The count can be higher than the changes in the change log. 999 is the maximum size and will roll back all the changes.

Example:

```
SQL> lb rollback syme_table.xml 999
lb rollback syme_table.xml 999
Table "Syme" dropped.
```

liquibase(lb) diff *<DEST URL>* *<DEST USER>* *<DEST PASS>* {report}

Displays differences between the current connection and the specified database. When the report is true, the output will be in the form of a text report. When the report is false, which is the default, the output will be in the form of a change log.

The destination URL format is HOST:PORT:SID or HOST:PORT/SERVICE.

Example:

```
SQL> lb diff localhost:1521/pdb1 hr2 hr2
lb diff localhost:1521/pdb1 hr2 hr2
Action successfully completed please review created file diffResult.xml

diffResult.xml
<?xml version="1.1" encoding="UTF-8" standalone="no"?>
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog-ext
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd
http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.6.xsd">
    <changeSet author="SKUTZ (generated)" id="1560183964715-1">
        <createTable tableName="DATABASECHANGELOG_EXPORT">
            <column name="RANK" type="NUMBER">
                <constraints nullable="false"/>
            </column>
            <column name="TYPE" type="VARCHAR2(100 BYTE)">
                <constraints primaryKey="true"
                primaryKeyName="DATABASECHANGELOG_EXPORT_PK"/>
            </column>
            <column name="SEQ" type="NUMBER">
                <constraints primaryKey="true"
                 primaryKeyName="DATABASECHANGELOG_EXPORT_PK"/>
            </column>
```

```
            <column name="META" type="CLOB">
                <constraints nullable="false"/>
            </column>
            <column name="OBJECT_NAME" type="VARCHAR2(200 BYTE)"/>
            <column name="FILE_NAME" type="VARCHAR2(200 BYTE)"/>
            <column name="DEP_NAME" type="VARCHAR2(200 BYTE)"/>
            <column name="DEP_COUNT" type="NUMBER"/>
        </createTable>
    </changeSet>
    <changeSet author="SKUTZ (generated)" id="1560183964715-2">
        <addColumn tableName="EMPLOYEES">
            <column name="FOOBAR" type="NUMBER"/>
        </addColumn>
    </changeSet>
    <changeSet author="SKUTZ (generated)" id="1560183964715-3">
        <createIndex indexName="ORDERED_ROWS" tableName="DATABASECHANGE-
LOG_EXPORT">
            <column name="RANK"/>
            <column name="SEQ"/>
        </createIndex>
    </changeSet>
    <changeSet author="SKUTZ (generated)" id="1560183964715-4">
        <createIndex indexName="TYPE_INDEX" tableName="DATABASECHANGE-
LOG_EXPORT">
            <column name="TYPE"/>
        </createIndex>
    </changeSet>
</databaseChangeLog>

SQL> lb diff localhost:1521/pdb1 hr2 hr2 true
lb diff localhost:1521/pdb1 hr2 hr2
Action successfully completed please review created file diffResult.txt

diffResult.txt
Reference Database: HR @ jdbc:oracle:thin:@(DESCRIPTION =    (ADDRESS_LIST
=
(ADDRESS = (PROTOCOL = TCP)(HOST = 0.0.0.0)(PORT = 1521))    )    (CON-
NECT_DATA =
(SERVICE_NAME = pdb1)    )  ) (Default Schema: HR)
Comparison Database: HR2 @ jdbc:oracle:thin:@localhost:1521/pdb1 (Default
Schema: HR2)
Compared Schemas: HR -> HR2
Product Name: EQUAL
Product Version: EQUAL
Missing Catalog(s): NONE
Unexpected Catalog(s): NONE
Changed Catalog(s): NONE
Missing Column(s):
     HR.DATABASECHANGELOG_EXPORT.DEP_COUNT
     HR.DATABASECHANGELOG_EXPORT.DEP_NAME
     HR.DATABASECHANGELOG_EXPORT.FILE_NAME
     HR.EMPLOYEES.FOOBAR
     HR.DATABASECHANGELOG_EXPORT.META
     HR.DATABASECHANGELOG_EXPORT.OBJECT_NAME
     HR.DATABASECHANGELOG_EXPORT.RANK
```

```
     HR.DATABASECHANGELOG_EXPORT.SEQ
     HR.DATABASECHANGELOG_EXPORT.TYPE
Unexpected Column(s): NONE
Changed Column(s): NONE
Missing Foreign Key(s): NONE
Unexpected Foreign Key(s): NONE
Changed Foreign Key(s): NONE
Missing Index(s):
     DATABASECHANGELOG_EXPORT_PK UNIQUE  ON HR.DATABASECHANGELOG_EX-
PORT(SEQ, TYPE)
     ORDERED_ROWS ON HR.DATABASECHANGELOG_EXPORT(RANK, SEQ)
     TYPE_INDEX ON HR.DATABASECHANGELOG_EXPORT(TYPE)
Unexpected Index(s): NONE
Changed Index(s): NONE
Missing Primary Key(s):
     DATABASECHANGELOG_EXPORT_PK on HR.DATABASECHANGELOG_EXPORT(SEQ, TYPE)
Unexpected Primary Key(s): NONE
Changed Primary Key(s): NONE
Missing Sequence(s): NONE
Unexpected Sequence(s): NONE
Changed Sequence(s): NONE
Missing Stored Procedure(s): NONE
Unexpected Stored Procedure(s): NONE
Changed Stored Procedure(s): NONE
Missing Table(s):
     DATABASECHANGELOG_EXPORT
Unexpected Table(s): NONE
Changed Table(s): NONE
Missing Unique Constraint(s): NONE
Unexpected Unique Constraint(s): NONE
Changed Unique Constraint(s): NONE
Missing View(s): NONE
Unexpected View(s): NONE
Changed View(s): NONE
```

liquibase(lb) status <*CHANGE LOG*>

Checks the status of the change log using the current connection. This shows if the change log has been applied and the result of the change log.

Example:

```
SQL> lb status syme_table.xmllb status syme_table.xml1 change
sets have not been applied to HR@jdbc:oracle:thin:@(DESCRIPTION =  (AD-
DRESS_LIST = (ADDRESS =
(PROTOCOL = TCP)(HOST = 0.0.0.0)(PORT =  1521)) ) (CONNECT_DATA = (SERV-
ICE_NAME = pdb1) )
) syme_table.xml::f3d9d927-3c5a-415c-a6fa-1dbc35a9da17::Generated
```

liquibase(lb) validate <*CHANGE LOG*>

Verifies if the change log is syntactically correct.

Example:

```
SQL> lb validate syme_table.xml
lb validate syme_table.xml

No issues were found in file syme_table.xml, validation passed.
```

liquibase(lb) changelogsync <*CHANGE LOG*>

Writes the SQL statements to mark all changes in the change log as executed in the database to a file.

Example:

```
SQL> lb changelogsync syme_table.xml
lb changelogsync syme_table.xml
Action successfully completed please review created file changelog-
sync_syme_table.xml
```

liquibase(lb) listlocks <*CHANGE LOG*>

Lists who currently has locks on the database change log.

Example:

```
SQL> lb listlocks syme_tbales.xml
lb listlocks syme_tbales.xml

ID     USER
!1     Skutz-laptop (10.39.200.228)
```

liquibase(lb) releaselocks <*CHANGE LOG*>

Releases all locks on the database change log.

Example:

```
SQL> lb releaselocks syme_table.xml
lb releaselocks syme_table.xml

Operation completed successfully all locks on syme_table.xml released
```

liquibase(lb) clearchecksums <*CHANGE LOG*>

Removes the current checksums from the database. In the next run, the checksums will be recomputed.

Example:

```
SQL> lb clearchecksums syme_table.xml
lb clearchecksums syme_table.xml

Operation completed successfully all checksums cleared for syme_table.xml
```

liquibase(lb) help <*COMMAND*>

Lists the available Liquibase commands. Type a command with no options for help on a specific command.

Example:

```
SQL> lb help

lb help
Provides a command line interface to Liquibase change management features
from within SQLcl!
Available commands are

liquibase(lb) genobject <object_type> <object_name>
liquibase(lb) genschema
liquibase(lb) gencontrolfile
liquibase(lb) update <CHANGE LOG> {include schema}
liquibase(lb) updatesql <CHANGE LOG> {include schema}
liquibase(lb) rollback <CHANGE LOG> <COUNT>
liquibase(lb) diff <DEST URL> <DEST USER> <DEST PASS> {report}
liquibase(lb) status <CHANGE LOG>
liquibase(lb) validate <CHANGE LOG>
liquibase(lb) changelogsync <CHANGE LOG>
liquibase(lb) listlocks <CHANGE LOG>
liquibase(lb) releaseLocks <CHANGE LOG>
liquibase(lb) clearchecksums <CHANGE LOG>
liquibase(lb) help <COMMAND>
```

2.5 ChangeSets

The following table lists the changeSets and provides a description for each of them. To learn more about changeSets, see <changeSet> tag.

ChangeSet	Description
CreateOracleConstraint	Creates a constraint from SQL.
CreateOracleFunction	Creates a function from SQL.
CreateOracleGrant	Creates a grant from SQL.
CreateOraclePackageBody	Creates a package body from SQL.
CreateOraclePackageSpec	Creates a package specification from SQL.
CreateOracleProcedure	Creates a procedure from SQL.
CreateOraclePublicSynonym	Creates a public synonym from SQL.
CreateOracleRefConstraint	Creates a referential constraint from SQL.
CreateOracleSynonym	Creates a synonym from SQL.
CreateOracleTrigger	Creates a trigger from SQL.
CreateOracleTypeBody	Creates a type body from SQL.
CreateOracleTypeSpec	Creates a type spec from SQL.
CreateSxmlObject	Creates a function from SQL.
DropOracleConstraint	Drops a constraint.

ChangeSet	Description
DropOracleFunction	Drops a function.
DropOracleGrant	Drops a grant.
DropOraclePackageBody	Drops a package body.
DropOraclePackageSpec	Drops a package specification.
DropOracleProcedure	Drops a procedure.
DropOracleRefConstraint	Drops a referential constraint.
DropOracleTrigger	Drops a trigger.
DropOracleTypeBody	Drops a type body.
DropOracleTypeSpec	Drops a type specification.
DropOracleSynonym	Drops a synonym.
DropSxmlObject	Drops an SXML object. If the object was created through createSxmlObject, this rolls back the object to the last state. If not created, the object is just dropped. This is primarily used internally for SXML object handling.
RunOracleScript	Runs an Oracle script.

2.6 Examples Using Liquibase

Some examples for using the Liquibase feature in SQLcl are:

- **Capture and Deploy an Object**
- **Capture and Deploy a Schema**
- **Generate the Master Control File**
- **Capture and Deploy a Schema and then Upgrade it and Redeploy**

Capture and Deploy an Object

To deploy the emp table from hr to hr2:

1. Connect to hr.

```
SQL> connect hr/hr
connect hr/hr
Connected.
```

2. Capture the object.

```
SQL> lb genobject table emp
lb genobject table emp
Action successfully completed please review created file emp_table.xml
```

3. Connect to the other user.

```
SQL> connect hr2/hr2
connect hr2/hr2
Connected.
```

4. Ensure the object does not already exist.

```
SQL> drop table emp
drop table emp
```

```
Table EMP dropped.
```

5. Create the object in the current schema.

> **Note:**
>
> As the schema name has changed, you must set *include schema* to `false` or it will try and create the object in the HR schema.

```
SQL> lb update emp_table.xml false
lb update emp_table.xml false

Table "EMP" created.
```

6. Verify the object was created.

```
SQL> desc emp
desc emp
Name Null? Type
---- ----- ------
ID NUMBER
```

Capture and Deploy a Schema

To capture the HR schema and reproduce it in the HR2 schema:

sql.exe hr/hr@pdb1

```
SQL> lb genschema
lb genschema
[Type - TYPE_SPEC]: 142 ms
[Type - TYPE_BODY]: 27 ms
[Type - SEQUENCE]: 61 ms
[Type - CLUSTER]: 25 ms
[Type - TABLE]: 447 ms
[Type - MATERIALIZED_VIEW_LOG]: 18 ms
[Type - MATERIALIZED_VIEW]: 6 ms
[Type - VIEW]: 143 ms
[Type - REF_CONSTRAINT]: 261 ms
[Type - DIMENSION]: 17 ms
[Type - FUNCTION]: 63 ms
[Type - PROCEDURE]: 66 ms
[Type - PACKAGE_SPEC]: 29 ms
[Type - DB_LINK]: 19 ms
[Type - SYNONYM]: 19 ms
[Type - INDEX]: 199 ms
[Type - TRIGGER]: 39 ms
[Type - PACKAGE_BODY]: 39 ms
[Method loadCaptureTable]: 1620 ms
[Method parseCaptureTableRecords]: 6433 ms
[Method sortCaptureTable]: 25 ms
[Method createExportChangeLogs]: 3 ms

Export Flags Used:
Export Grants false
Export Synonyms false

SQL> connect system/sparrow
connect system/sparrow
```

```
Connected.

setup the hr2 user --

drop user hr2 cascade;
create user hr2 identified by hr2;
grant connect,resource, create view to hr2;
alter user hr2 quota unlimited on users;
alter user hr2 quota unlimited on sysaux;

SQL> connect hr2/hr2
connect hr2/hr2
Connected.

SQL> lb update controller.xml
lb update controller.xml

Sequence "DEPARTMENTS_SEQ" created.

Sequence "LOCATIONS_SEQ" created.

Sequence "EMPLOYEES_SEQ" created.

Table "COUNTRIES" created.

Table "REGIONS" created.

Table "LOCATIONS" created.

Table "DEPARTMENTS" created.

Table "JOBS" created.

Table "EMPLOYEES" created.

Table "JOB_HISTORY" created.

Table "Syme" created.

Table "FOO" created.

View "EMP_DETAILS_VIEW" created.

Table "COUNTRIES" altered.

Table "LOCATIONS" altered.

Table "DEPARTMENTS" altered.

Table "EMPLOYEES" altered.

Table "EMPLOYEES" altered.

Table "EMPLOYEES" altered.

Table "DEPARTMENTS" altered.

Table "JOB_HISTORY" altered.

Table "JOB_HISTORY" altered.
```

```
Table "JOB_HISTORY" altered.

Function FUNCTION1 compiled

Procedure SECURE_DML compiled

Procedure ADD_JOB_HISTORY compiled

Procedure PROCEDURE1 compiled

Index "LOC_COUNTRY_IX" created.

Index "JHIST_JOB_IX" created.

Index "LOC_STATE_PROVINCE_IX" create.

Index "EMP_DEPARTMENT_IX" created.

Index "JHIST_EMPLOYEE_IX" created.

Index "LOC_CITY_IX" created.

Index "JHIST_DEPARTMENT_IX" created.

Index "EMP_JOB_IX" created.

Index "EMP_MANAGER_IX" created.

Index "EMP_NAME_IX" created.

Index "DEPT_LOCATION_IX" created.

Trigger SECURE_EMPLOYEES compiled

Trigger "SECURE_EMPLOYEES" altered.

Trigger UPDATE_JOB_HISTORY compiled
```

Generate the Master Control File

```
SQL> lb gencontrolfile
lb gencontrolfile
Action successfully completed please review created file controller.xml

SQL> !type controller.xml
!type controller.xml
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd">
<include file="{filename.xml}"/>
</databaseChangeLog>
```

Capture and Deploy a Schema and then Upgrade it and Redeploy

1. Migrate the HR schema to the HR2 schema, then modify the HR schema and update HR2.

```
Drop/create HR2 user as system

SQL> connect system/sparrow
connect system/sparrow
Connected.
SQL> drop user hr2 cascade;

User HR2 dropped.

SQL> create user hr2 identified by hr2;

User HR2 created.

SQL> grant connect,resource, create view to hr2;

Grant succeeded.

SQL> alter user hr2 quota unlimited on users;

User HR2 altered.

SQL>
alter user hr2 quota unlimited on sysaux;

User HR2 altered.

SQL> !mkdir v1
!mkdir v1

SQL> cd v1
cd v1

SQL> connect hr/hr
connect hr/hr
Connected.

SQL> lb genschema
lb genschema
[Type - TYPE_SPEC]: 163 ms
[Type - TYPE_BODY]: 30 ms
[Type - SEQUENCE]: 57 ms
[Type - CLUSTER]: 27 ms
[Type - TABLE]: 311 ms
[Type - MATERIALIZED_VIEW_LOG]: 19 ms
[Type - MATERIALIZED_VIEW]: 8 ms
[Type - VIEW]: 155 ms
[Type - REF_CONSTRAINT]: 293 ms
[Type - DIMENSION]: 29 ms
[Type - FUNCTION]: 77 ms
[Type - PROCEDURE]: 65 ms
[Type - PACKAGE_SPEC]: 31 ms
[Type - DB_LINK]: 13 ms
[Type - SYNONYM]: 18 ms
[Type - INDEX]: 188 ms
[Type - TRIGGER]: 41 ms
[Type - PACKAGE_BODY]: 37 ms
[Method loadCaptureTable]: 1562 ms
[Method parseCaptureTableRecords]: 8437 ms
[Method sortCaptureTable]: 37 ms
[Method createExportChangeLogs]: 4 ms
```

```
Export Flags Used:
Export Grants false
Export Synonyms false
```

2. Modify the HR Schema.

```
SQL> ALTER TABLE COUNTRIES ADD (COLUMN1 VARCHAR2(20) );

Table COUNTRIES altered.

SQL> ALTER TABLE "Syme" ADD (COLUMN1 VARCHAR2(20) );

Table "Syme" altered.

SQL> ALTER TABLE "Syme" ADD (COLUMN2 VARCHAR2(20) );

Table "Syme" altered.

SQL> ALTER TABLE "Syme" ADD (COLUMN3 VARCHAR2(20) );

Table "Syme" altered.

SQL> CREATE VIEW VIEW1 AS SELECT * FROM "Syme";

View VIEW1 created.

SQL> CREATE OR REPLACE PROCEDURE PROCEDURE2 (PARAM1 IN VARCHAR2,PARAM2 IN VAR-
CHAR2,PARAM3 IN VARCHAR2, PARAM4 IN VARCHAR
2) AS BEGIN NULL; END PROCEDURE2;

Procedure PROCEDURE2 compiled
```

3. Make a change to the v2 directory.

```
SQL> cd ..
cd ..
SQL> !mkdir v2
!mkdir v2

SQL> cd v2
cd v2
S
```

4. Export v2 of the HR schema.

```
SQL> lb genschema
lb genschema
[Type - TYPE_SPEC]: 44 ms
[Type - TYPE_BODY]: 30 ms
[Type - SEQUENCE]: 51 ms
[Type - CLUSTER]: 26 ms
[Type - TABLE]: 447 ms
[Type - MATERIALIZED_VIEW_LOG]: 13 ms
[Type - MATERIALIZED_VIEW]: 6 ms
[Type - VIEW]: 110 ms
[Type - REF_CONSTRAINT]: 258 ms
[Type - DIMENSION]: 16 ms
[Type - FUNCTION]: 56 ms
[Type - PROCEDURE]: 62 ms
[Type - PACKAGE_SPEC]: 29 ms
[Type - DB_LINK]: 12 ms
[Type - SYNONYM]: 13 ms
[Type - INDEX]: 205 ms
```

```
[Type - TRIGGER]: 39 ms
[Type - PACKAGE_BODY]: 34 ms
[Method loadCaptureTable]: 1451 ms
[Method parseCaptureTableRecords]: 7015 ms
[Method sortCaptureTable]: 23 ms
[Method createExportChangeLogs]: 4 ms

Export Flags Used:
Export Grants false
Export Synonyms false
```

5. To import v1, change back to the v1 directory and connect as HR2.

```
SQL> cd ../v1
cd ../v1
SQL> connect hr2/hr2
connect hr2/hr2
Connected.
```

6. Now import the capture into the HR2 schema.

```
SQL> lb update controller.xml
lb update controller.xml
Sequence "DEPARTMENTS_SEQ" created.
Sequence "LOCATIONS_SEQ" created.
Sequence "EMPLOYEES_SEQ" created.
Table "COUNTRIES" created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "REGIONS" created.
Table "LOCATIONS" created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "DEPARTMENTS" created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "JOBS" created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "EMPLOYEES" created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
```

```
Comment created.
Comment created.
Table "JOB_HISTORY" created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "Syme" created.
Table "FOO" created.
View "EMP_DETAILS_VIEW" created.
Table "COUNTRIES" altered.
Table "LOCATIONS" altered.
Table "DEPARTMENTS" altered.
Table "EMPLOYEES" altered.
Table "EMPLOYEES" altered.
Table "EMPLOYEES" altered.
Table "DEPARTMENTS" altered.
Table "JOB_HISTORY" altered.
Table "JOB_HISTORY" altered.
Table "JOB_HISTORY" altered.
Function FUNCTION1 compiled
Procedure SECURE_DML compiled
Procedure ADD_JOB_HISTORY compiled
Procedure PROCEDURE1 compiled
Index "LOC_COUNTRY_IX" created.
Index "JHIST_JOB_IX" created.
Index "LOC_STATE_PROVINCE_IX" created.
Index "EMP_DEPARTMENT_IX" created.
Index "JHIST_EMPLOYEE_IX" created.
Index "LOC_CITY_IX" created.
Index "JHIST_DEPARTMENT_IX" created.
Index "EMP_JOB_IX" created.
Index "EMP_MANAGER_IX" created.
Index "EMP_NAME_IX" created.
Index "DEPT_LOCATION_IX" created.
Trigger SECURE_EMPLOYEES compiled
Trigger "SECURE_EMPLOYEES" altered.
Trigger UPDATE_JOB_HISTORY compiled
```

7. Check a table to verify.

```
SQL> desc "Syme"
desc "Syme"
Name Null? Type
---- ----- ------
ID NUMBER
SQL> desc countries
desc countries
Name Null? Type
------------ -------- ------------
COUNTRY_ID NOT NULL CHAR(2)
COUNTRY_NAME VARCHAR2(40)
REGION_ID NUMBER
```

8. Switch to v2 and import the capture.

```
SQL> cd ../v2
cd ../v2
SQL> lb update controller.xml
lb update controller.xml
Table "HR2"."COUNTRIES" altered.
```

```
Table "HR2"."COUNTRIES" altered.
Comment created.
Comment created.
Comment created.
Comment created.
Table "HR2"."LOCATIONS" altered.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "HR2"."DEPARTMENTS" altered.
Table "HR2"."DEPARTMENTS" altered.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "HR2"."EMPLOYEES" altered.
Table "HR2"."EMPLOYEES" altered.
Table "HR2"."EMPLOYEES" altered.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "HR2"."JOB_HISTORY" altered.
Table "HR2"."JOB_HISTORY" altered.
Table "HR2"."JOB_HISTORY" altered.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Comment created.
Table "HR2"."Syme" altered.
Table "HR2"."Syme" altered.
Table "HR2"."Syme" altered.
View "VIEW1" created.
Table "COUNTRIES" altered.
Table "LOCATIONS" altered.
Table "DEPARTMENTS" altered.
Table "EMPLOYEES" altered.
Table "EMPLOYEES" altered.
Table "EMPLOYEES" altered.
Table "DEPARTMENTS" altered.
Table "JOB_HISTORY" altered.
```

```
Table "JOB_HISTORY" altered.
Table "JOB_HISTORY" altered.
Function FUNCTION1 compiled
Procedure SECURE_DML compiled
Procedure ADD_JOB_HISTORY compiled
Procedure PROCEDURE1 compiled
Procedure PROCEDURE2 compiled
Trigger SECURE_EMPLOYEES compiled
Trigger UPDATE_JOB_HISTORY compiled
```

9. Verify that you are on v2.

```
SQL> desc "Syme"
desc "Syme"
Name Null? Type
------- ----- ------------
ID NUMBER
COLUMN1 VARCHAR2(20)
COLUMN2 VARCHAR2(20)
COLUMN3 VARCHAR2(20)
SQL> desc countries
desc countries
Name Null? Type
------------ -------- ------------
COUNTRY_ID NOT NULL CHAR(2)
COUNTRY_NAME VARCHAR2(40)
REGION_ID NUMBER
COLUMN1 VARCHAR2(20)
```