



# Taking Batch Scripting to the Next Level with SQLcl

21<sup>st</sup> September 2017

**Robert Marz**  
Technical Architect

**DOAG**



# Robert Marz

Client

Senior Technical Architect  
with database centric view of the world

its-people

Portfolio Manager Database Technologies  
Blog Editor

DOAG

Active Member Database Community  
in charge of Cloud topics



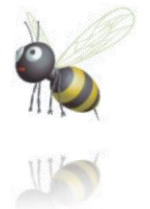
@RobbieDatabee



blog.its-people.de



Robert.Marz  
@its-people.de



# What's in Store for you?

*Scripts and Slides  
available for  
Download*



## What to expect:

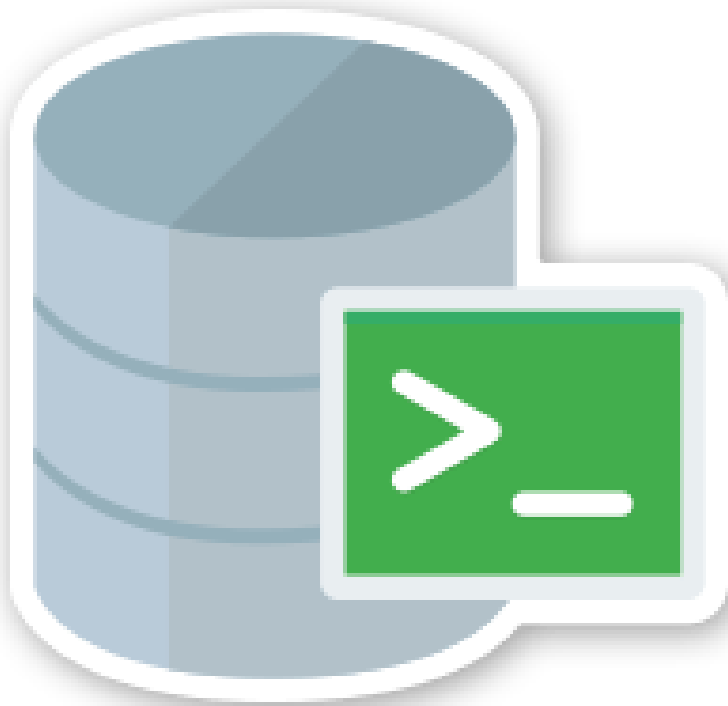
- Discover new Possibilities
- Examples



## Not included:

- Introduction to JavaScript
- Complete Feature Overview

# What is SQLcl?



## Oracle SQL Developer Command Line

- The new SQL\*Plus
- Modern Command Line
- Production Release since OOW
- Included in 12cR2  
\$ORACLE\_HOME/bin

# Scripting Vintage Style



SQL\*Plus

stable for decades

Sequential  
SQL & PL/SQL

Not too dynamic  
„DEFINE“

Error Handling  
„whenever sqlerror“

# Scripting in SQLcl

Languages  
(JSR-223)

New  
Possibilities

SQL\*Plus  
„plus“

# Scripting in SQLcl – Basics

## GitHub Readme

<https://github.com/oracle/oracle-db-tools/blob/master/sqlcl/README.md>

## ctx

```
ctx.write(<string>)
print();
```

## sqlcl

```
sqlcl.setStmt(<string>)
sqlcl.run()
```

## util

```
execute(<string>,binds)
executeReturnOneCol(<string>,binds)
executeReturnListofLists(<string>,binds)
executeReturnList(<string>,binds)
```

## Globals

There are a few globals pushed into the scripting engine for use.

**args** - This is a simple array of the arguments passed along

Example:

```
for(var arg in args) {
  ctx.write(arg + ":" + args[arg]);
  ctx.write("\n");
}
```

**sqlcl** - This is SQLCL itself

```
setStmt(<String of stuff to run>)
  This can be a single statement, an entire script of stuff, or any sqlcl command such as "@numbers.sql"
```

```
run()
  Runs whatever is set via the setStmt function
```

Example:

```
/* Run any amount of command in the sqlcl prompt */
sqlcl.setStmt("select something from somewhere; @myscript \n begin null;end;");

sqlcl.run();
```

**ctx** ( this has tons of methods but this is the single most important )

```
write(<String>)
```

Example:

```
ctx.write('Hello World');
```

**util** ( again tons of methods )

```
execute(<string>,binds)
  executes whatever is passed in with a boolean return for success/failure
```

```
executeReturnOneCol(<string>,binds)
  executes and returns the first row , first column
```

```
executeReturnListofList(<string>,binds)
  executes and returns an array(rows) of arrays(row).
```

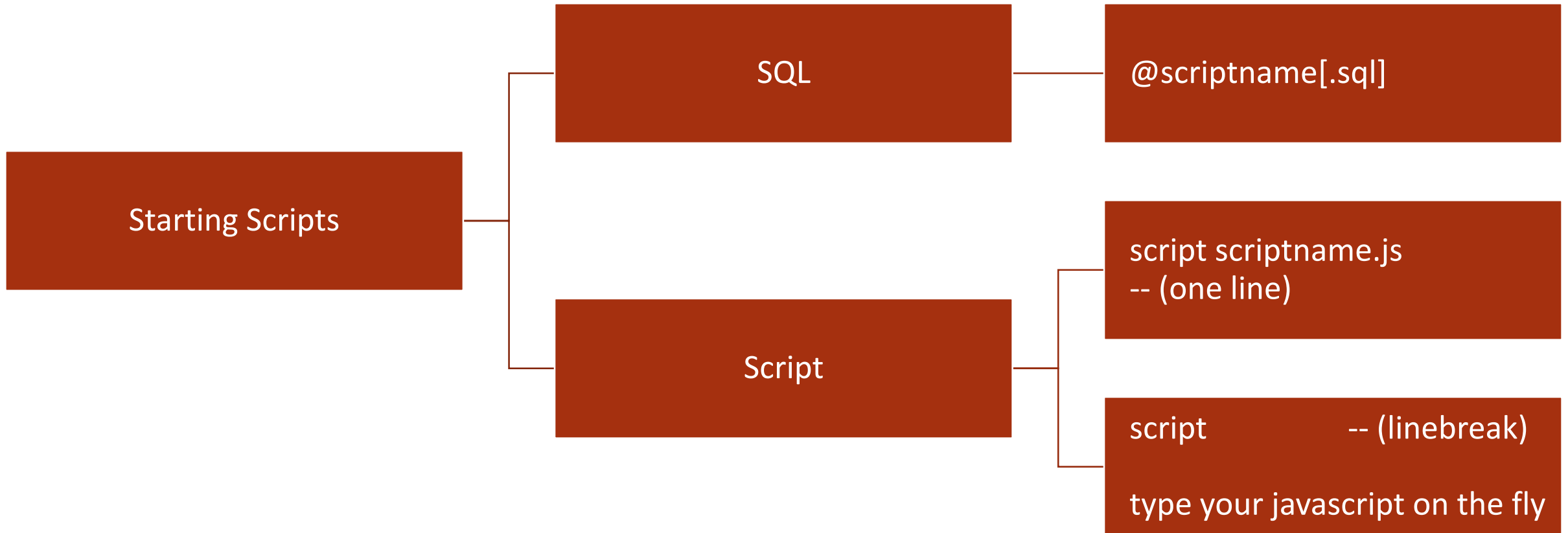
```
executeReturnList(<string>,binds)
  execute and returns and array ( rows ) of objects ( row )
```

# SQLcl – Warm-Up

Invoking

```
sql usr[/pwd[@connect]] [as sysdba]
```

# SQLcl – Warm-Up



# SQLcl – Warm-Up – Demo 01.1

---

SQLcl  
Persistence

Objects outlast the End of  
the Script

---

Demo Object Persistence

Assign Variables

---

Access from next Script

---

# SQLcl – Warm-Up – Demo 01.1 cont.

```
-- First Script
script
  // Use Java Class to access Properties of SQLcl
  var myHistory = Java.type("oracle.dbtools.raptor.console.MultiLineHistory");
  var dontForget : -- Do some SQL stuff; Next Script
  var historyMaxS script
  function printC printCurrentHistory(myHistory);
    print("Cur print("Cut
      SQLclHi myHistory.getInstance().setMaxSize(20);
  } printCurrentHistorySize(myHistory);
/
```

Demo

```
-- Next Script
script
  print(dontForget);
/
```

# SQLcl – Warm-Up – Demo 01.2

---

## Alias

Suitable for small, reoccurring tasks

---

May contain SQL, PL/SQL or Script

---

Definition is persistent

---

## Demo

Restore Data + RememberMe

---

# SQLcl – Warm-Up – Demo 01.2 cont.

```
alias restore_emp=begin
  delete from employees;
  insert into employees
    select * from employees_backup;
  commit;
end;
/
alias remindMe=script
  if(typeof dontForget != 'undefined') {
    print("Don't forget: "+ dontForget)
  }
;
```

# SQLcl – Warm-Up – Demo 01.3

---

login.sql    Customise your command line  
using JavaScript

---

SQLPATH

set

---

show

---

Demo

Switching between  
prompt layouts

---

# SQLcl – Warm-Up – Demo 01.3 cont.

```
script
```

```
var dbUser = util.executeReturnC  
if ( dbUser == 'GEOFF' ) {  
  sqlcl.setStmt('set sqlprompt "@  
} else {  
  sqlcl.setStmt('set sqlprompt "@  
}  
  
sqlcl.run();
```

```
/
```

```
01 Basics> sql geoff/geoff  
SQLcl: Release 4.2.0 Production auf Di  
Copyright (c) 1982, 2017, Oracle. All @ " ");  
Last Successful login time: Di Feb 07 " ");  
Verbunden mit:  
Oracle Database 12c Enterprise Edition  
With the Partitioning, OLAP, Advanced  
Go GEOFF! connect chris/chris  
Angemeldet.  
Go CHRIS!
```

# SQLcl – Warm-Up – Demo 01.4

---

## Output Streams

`ctx.write()`

Needs trailing "\n"  
to produce output

---

`print()`

Works like expected

---

Uses Different output Stream

---

Demo

Mixed `ctx.write()` & `print()`  
statements

---

# SQLcl – Warm-Up – Demo 01.4 cont.

```
ctx.write( '1 - ctx\n' );  
print(    '2 - print' );  
ctx.write( '3 - c  
print(    '4 - p  
ctx.write( '5 - ctx\n' );  
print(    '6 - print' );
```

**Demo**

```
2 - print  
4 - print  
6 - print  
1 - ctx  
3 - ctx  
5 - ctx  
SQL>
```

# SQLcl – Flow Control – Demo 02

---

## SQL\*Plus

Sequential SQL Blocks

DEFINE, VARIABLE

Workarounds

Spool „new Script.sql“

PL/SQL execute immediate

---

## SQLcl

Modern Scripting Languages

JDBC Connection

Bind Variables

Result Evaluation

Access to SQLcl Interpreter

Dynamic Loading of Scripts

---

# SQLcl – Flow Control – Demo 02 (cont.)

---

Conditions    If – then – else

---

Case

---

Demo

Create table if not exists

---

---

Loops        For / While

---

Demo

Looping through Results

---

Assembling Script-Calls

---

# SQLcl – Flow Control – Demo 02a If-Then-Else

```
// Test existence of DB Objects
/* is the table already there? */
if (tabCnt == 0){
  print("Table " + tabName + " not found. Creating it...");

  sqlcl.setStmt( "set echo on\n"
    + "set feedback on \n"
    + "create table "+tabName+" number not null \n"
    + "
    + "
    + "
    );
  + "set serveroutput on size unl \n" // Everything that works in SQLcl can be used.
  + "alter table "+tabName+" add constraint pk_"+tabName+" primary key (indx);");
  // You will find all statements in the SQLcl History afterwards...
  sqlcl.run();
} else {
  ctx.write("Table "+tabName+" already exists.\n");
}
```

**Demo**

# SQLcl – Flow Control – Demo 02b Loops

```
// Loops
var binds = {};
var a = 0, b = 1, f = 1,
// Constructing Scripts
var stmt = "commit; \n";
binds={};
var ret = util.executeReturnL
for (i = 0; i < ret.length; i
    stmt = stmt + "@02_demo.s
}

ctx.write ("Script: \n"+stmt);
sqlcl.setStmt( stmt );
sqlcl.run();

-- 02_demo.sql
-- Print out the first two parameters
set verify off
set heading off
select &1, &2 as para2
from dual;

    .im <=3", binds);
```

Demo

# SQLcl – Loading Blobs – Demo 03

---

**Blob**

Transfer Blob from File  
into Database

---

Use Java Classes

Access Files via  
`java.nio.files.Files`

---

Demo

Copy Image to Table

---

# SQLcl – Loading Blobs – Demo 03

```
//script "03-LoadingBlobs.js" "<complete image path>"  
//script "03-LoadingBlobs.js" /home/oracle/image.jpg
```

```
if(!util.execute( "insert into dokument (datei_inhalt,datei_pfad, datum) values(:inhalt, :pfad,  
sysdate)"  
                , bindmap)  
) { print("insert failed, exiting");  
    exit;  
}  
sqlcl.setStmt( "commit; \n"  
              + "set sqlformat ansiconsole \n"  
              + ' select datei_pfad "Dateipfad",dbms_lob.getlength(datei_inhalt) "Größe", to_char(  
datum,\'DD.MM.YYYY HH24:MI:SS\') "Zeit" '  
              + "from dokument;");  
sqlcl.run();
```

**Demo**

```
bindmap.put("inhalt", blob);  
bindmap.put("pfad", filePath);
```

# SQLcl – Array Magic – Demo 04

---

Arrays

Powerful structures

Simple Types

---

Objects

---

Nested Arrays

---

Demo  
Deploy Framework

List of Credentials

---

Connection Check

---

Execute Scripts with Credentials

---

# SQLcl – Array Magic – Demo 04a – SQL-Define

```

-- define passwords
DEFINE DATA_PWD
DEFINE DBGDI_PWD
DEFINE DBIS_EXPORT_OWD_PWD
DEFINE DOOTZ_PWD
DEFINE EBA_LAERM_EXPORT_OWNER_PWD
DEFINE EBA_PLATZ_PWD
DEFINE ELBEKA_PWD
DEFINE EXPORT_PWD
DEFINE GDOSYS_PWD
DEFINE GEOBEANS_PWD
DEFINE GEOSAP_DATA_OWNER_PWD
DEFINE GEOSAP_META_OWNER_PWD
DEFINE GINA_OWNER_PWD
DEFINE ISK_2010_PWD
DEFINE ISK_DATA_OWNER_PWD
DEFINE ISK_DBGDI_TEST_PWD
DEFINE ISK_EXPORT_PWD
DEFINE ISK_GEO_OWNER_PWD
DEFINE ISK_VISU_EXT_PWD
DEFINE ISK_VISU_STAGE_TMP_PWD
DEFINE ISR_GEO_OWNER_PWD
DEFINE ISR_PWD

-- Test passwords
@&INSTALLER_CORE_PATH/process_check_connect.sql "&DATA."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&DBGDI."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&DBIS_EXPORT_OWD."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&DOOTZ."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&EBA_LAERM_EXPORT_OWNER."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&EBA_PLATZ."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ELBEKA."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&EXPORT."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&GDOSYS."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&GEOBEANS."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&GEOSAP_DATA_OWNER."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&GEOSAP_META_OWNER."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&GINA_OWNER."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_2010."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_DATA_OWNER."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_DBGDI_TEST."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_EXPORT."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_GEO_OWNER."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_VISU_EXT."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISK_VISU_STAGE_TMP."
@&INSTALLER_CORE_PATH/process_check_connect.sql "&ISR."

```

# SQLcl – Array Magic – Demo 04b – JS Arrays

```

// Users could be filled
users=[
  {"user" : "DATA", "password" : "E_sXxXx7o_w" },
  {"user" : "DBGD", "password" : "E_sXxXx7o_w" },
  {"user" : "DBIS", "password" : "E_sXxXx7o_w" },
  {"user" : "DOOT", "password" : "E_sXxXx7o_w" },
  {"user" : "BARR", "password" : "E_sXxXx7o_w" },
  {"user" : "EBA", "password" : "E_sXxXx7o_w" },
  {"user" : "EBA", "password" : "E_sXxXx7o_w" },
  {"user" : "ELBE", "password" : "E_sXxXx7o_w" },
  {"user" : "EXPO", "password" : "E_sXxXx7o_w" },
  {"user" : "GDOS", "password" : "E_sXxXx7o_w" },
  {"user" : "GEOB", "password" : "E_sXxXx7o_w" },
  {"user" : "GEOS", "password" : "E_sXxXx7o_w" },
  {"user" : "GEOSAP_META", "password" : "E_sXxXx7o_w" },
  {"user" : "GINA_OWNER", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_2010", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_DATA_OWNER", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_DBGDI_TEST", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_EXPORT", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_GEO_OWNER", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_VISU_EXT", "password" : "E_sXxXx7o_w" },
  {"user" : "ISK_VISU_STAGE_TMP", "password" : "E_sXxXx7o_w" },
  {"user" : "ISR_GEO_OWNER", "password" : "E_sXxXx7o_w" },
  {"user" : "ISR", "password" : "E_sXxXx7o_w" }
];

// Testing all Users
// Searching Password for user
function getPasswd(username){
  return users.filter(function ( obj ) {
    return obj.user === username;
  })[0];
}

// Using in Scripts:
var uname = "BARRY";
ctx.write( "The password of Schema "+uname+" is "
+ getPasswd(uname)+ "\n");

```

**Demo**

# SQLcl – Remote Control – Demo 05

---

Pipes

Copy Data

No Export File or DB Link

```
SQL> select /*insert*/
 2      employee_id, first_name, last_name, salary
 3      from geoff.employees
 4      where rownum <4;
REM INSERTING into GEOFF.EMPLOYEES
SET DEFINE OFF;
Insert into GEOFF.EMPLOYEES (EMPLOYEE_ID,FIRST_NAME, LAST_NAME, SALARY) values ('220','Curtis','Newton','6300');
Insert into GEOFF.EMPLOYEES (EMPLOYEE_ID,FIRST_NAME, LAST_NAME, SALARY) values ('219','Andreas','von der Meden','13000');
Insert into GEOFF.EMPLOYEES (EMPLOYEE_ID,FIRST_NAME, LAST_NAME, SALARY) values ('218','Marcus','Brody','2700');

SQL>
```

Demo

Copy Data between two  
Databases

# SQLcl – Background Sessions – „noDemo“ 06

---

Parallelise  
Tasks

Java Threads

GitHub  
oracle-db-tools

bg.js

longops.js

---

its-people Blog

yet to be written

---

Additional  
JDBC  
connections

Connection

metadata available  
except credentials  
SQLcl connection can  
be cloned

---

„n“ threads

# SQLcl – As you like it – Demo 07

---

JSR-223

e.g. JavaScript,  
Lua or Python

Java Implementations  
„nashorn“, „LuaJ“, „Jython“

---

Embedding SQLcl

Use SQLcl JARs in your Apps

---

Demo

Running lua & python from SQLcl

---

Using SQLcl from python

---

ECMA Script 5 & 6

---

# SQLcl – As you like it – Demo 07a

```
print ("Fibonacci first 10 in Lua (LuaJ)")
print "Fibonacci"
a, b = echo "Pointing the CLASSPATH to the interpreters"
while export CLASSPATH="jython-standalone-2.7.0.jar:lua-jse-3.0.1.jar"
do pr echo CLASSPATH: $CLASSPATH
a, sql -s geoff/geoff << EOF
script ./fibonacci.py
script ./fibonacci.lua
EOF
```



Demo

# SQLcl – As you like it – Demo 07c

```
/*  
cons export PATH=~/.jre-9/bin:$PATH  
cons export JAVA_HOME=/home/oracle/jre-9  
cons export APP_VM_OPTS="-Dnashorn.args=--language=es6"  
let java -version  
sql -s geoff/geoff << EOF  
script demo07c.js  
EOF
```



**Demo**

# The JavaScript

The JavaScript  
Edvard Munch  
1910



# Reading List

Getting  
Started

[SQLcl Scripting: Docs](#)

[SQLcl Community](#)

Basics

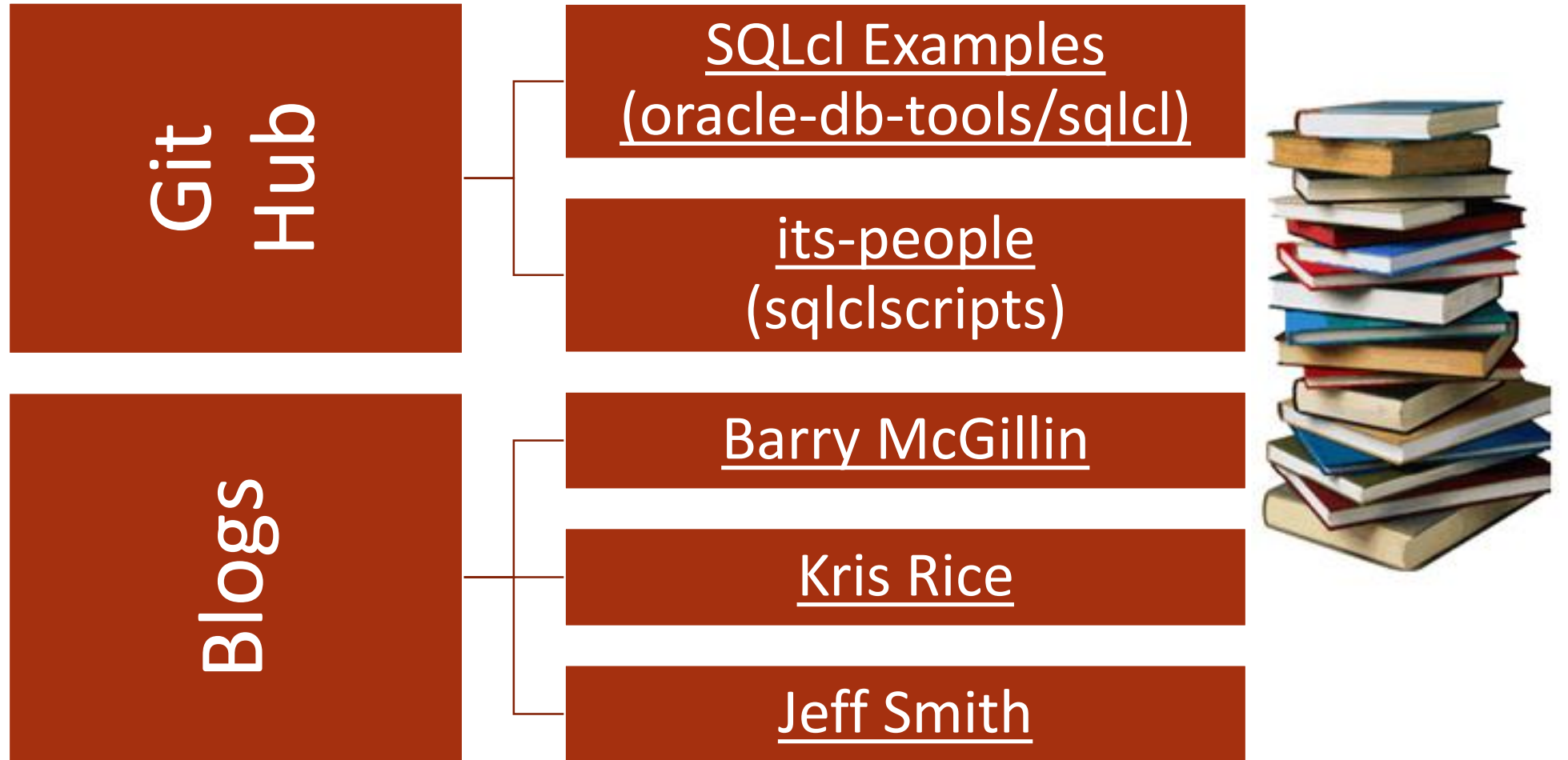
[List of JVM languages](#)

[JavaScript Cheat Sheet](#)

[Nashorn Tutorial](#)



# Reading List



# Conclusion

