Presented by,

MySQL AB® & O'Reilly Media, Inc.

MySQL Proxy
The complete tutorial

Jan Kneschke,

Senior software Engineer,

MySQL AB

Giuseppe Maxia,

Community Team Leader,

MySQL AB

Some assessment questions

- Who has used MySQL Proxy?
- Who has read the "getting started" article?
- Who has read the datacharmer blog?
- Who uses MySQL Proxy in production?

UNION INSERT CREATE DATABASE DOUBLE
REPLACE SELECT KILL DROP FLUSH
LOAD MODIFY OPTIMIZE DELETE EXIT

Agenda

- Overview
- Basic principles
- Lua scripts
- Proxy for a single backend
- Proxy for multiple backends
- Wizardry (all over)

DOWNLOAD MATERIAL

Slides and example scripts

http://datacharmer.org/tutorial_uc2008/

Proxy (< latin "procuratio")

= Someone taking care of someone else's interests

A server proxy is something acting on behalf of another server

Database problems

- broken?
- missing feature?
- not flexible?

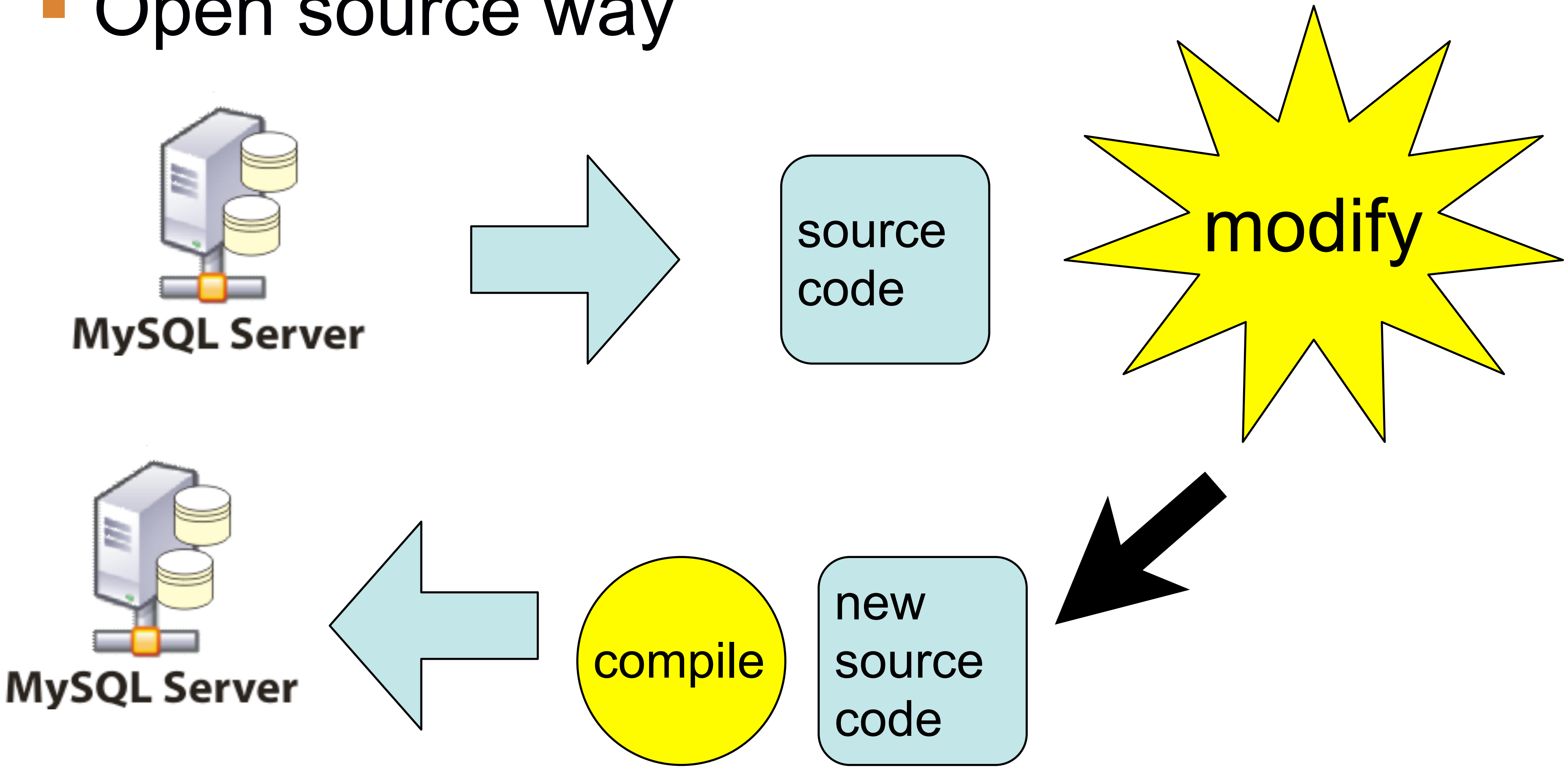
Solving database problems

- traditional way

1. file a bug report
2. wait

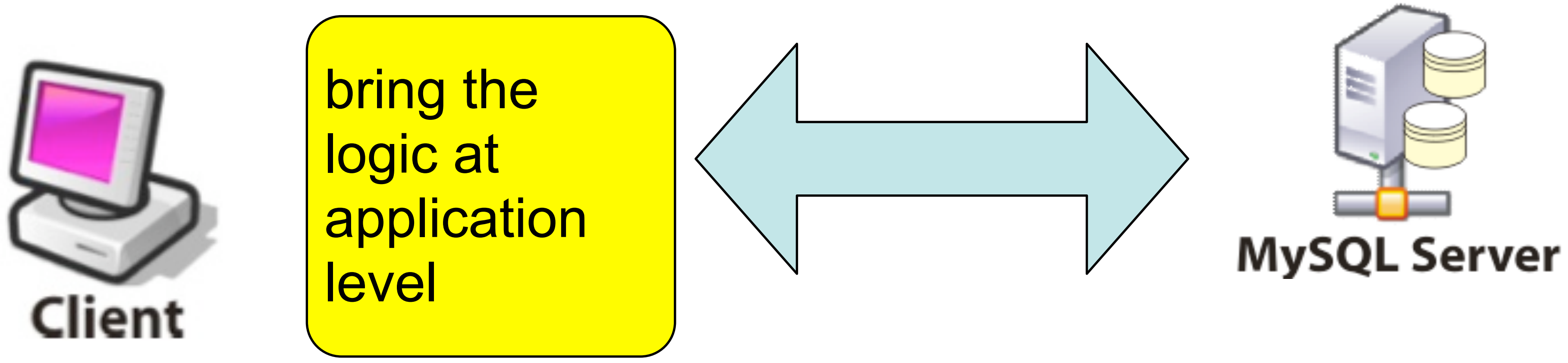
Solving database problems

- Open source way



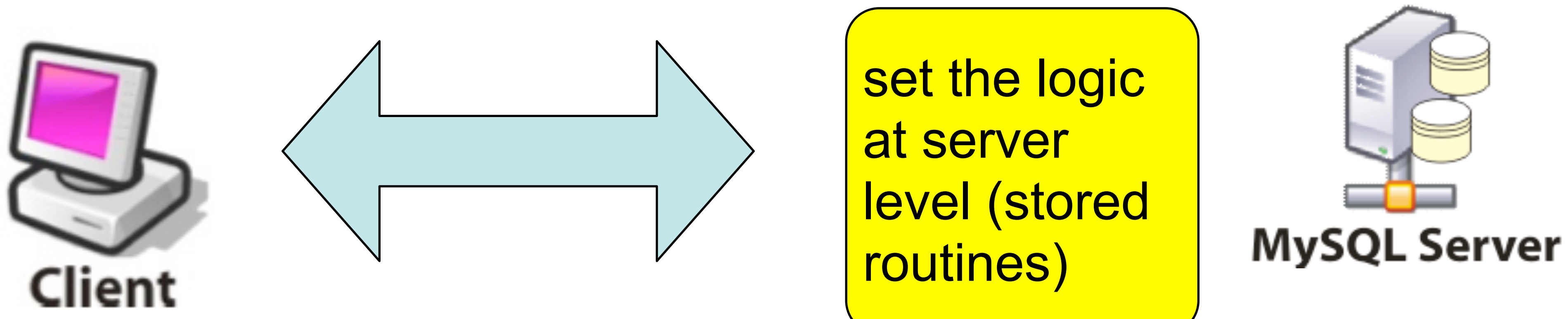
Solving database problems

- creative (shortsighted) way



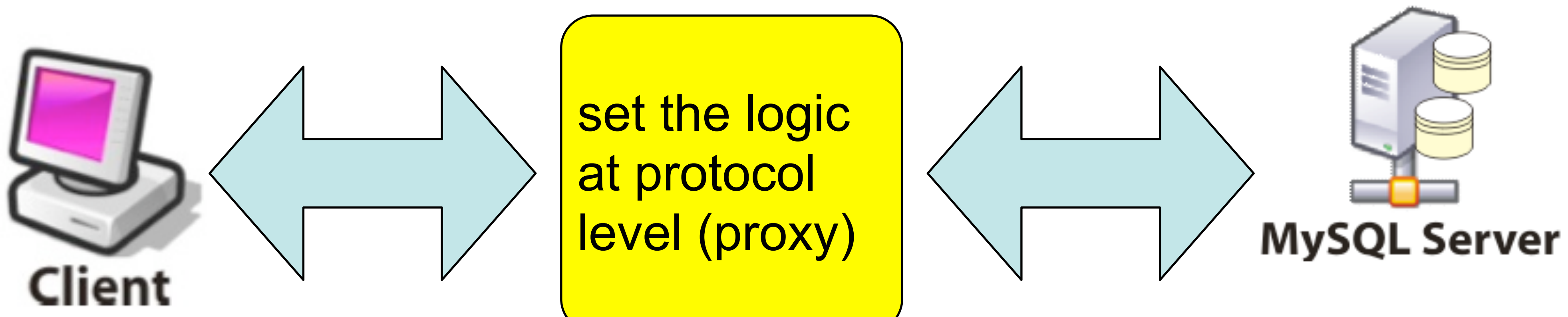
Solving database problems

- creative (enlightened) way



Solving database problems

- creative (more enlightened) way



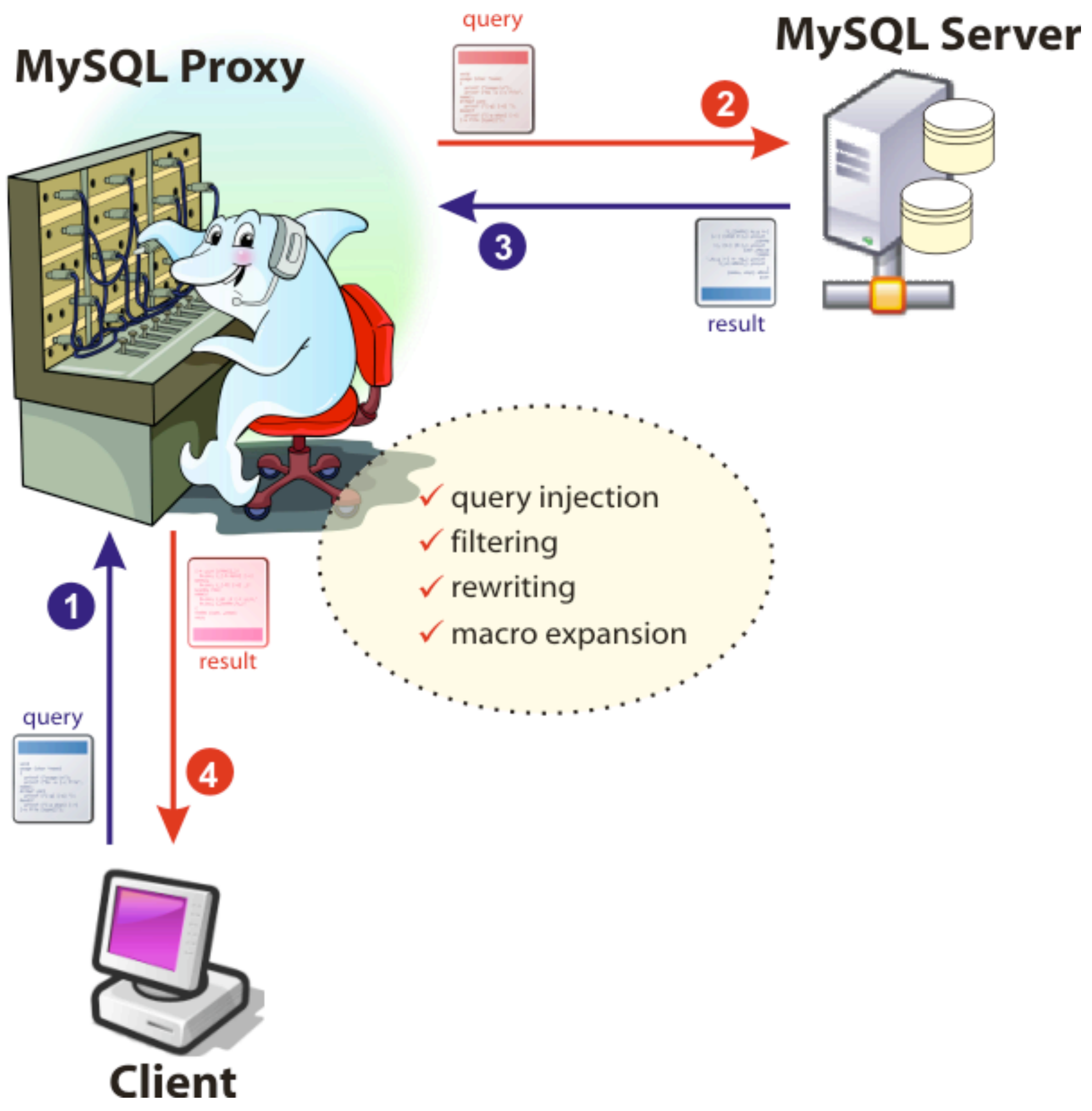
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
- start/stop a MySQL server remotely
- play movies (seriously!)
- make coffee (now, you're kidding, right? nope)
- load balancing servers

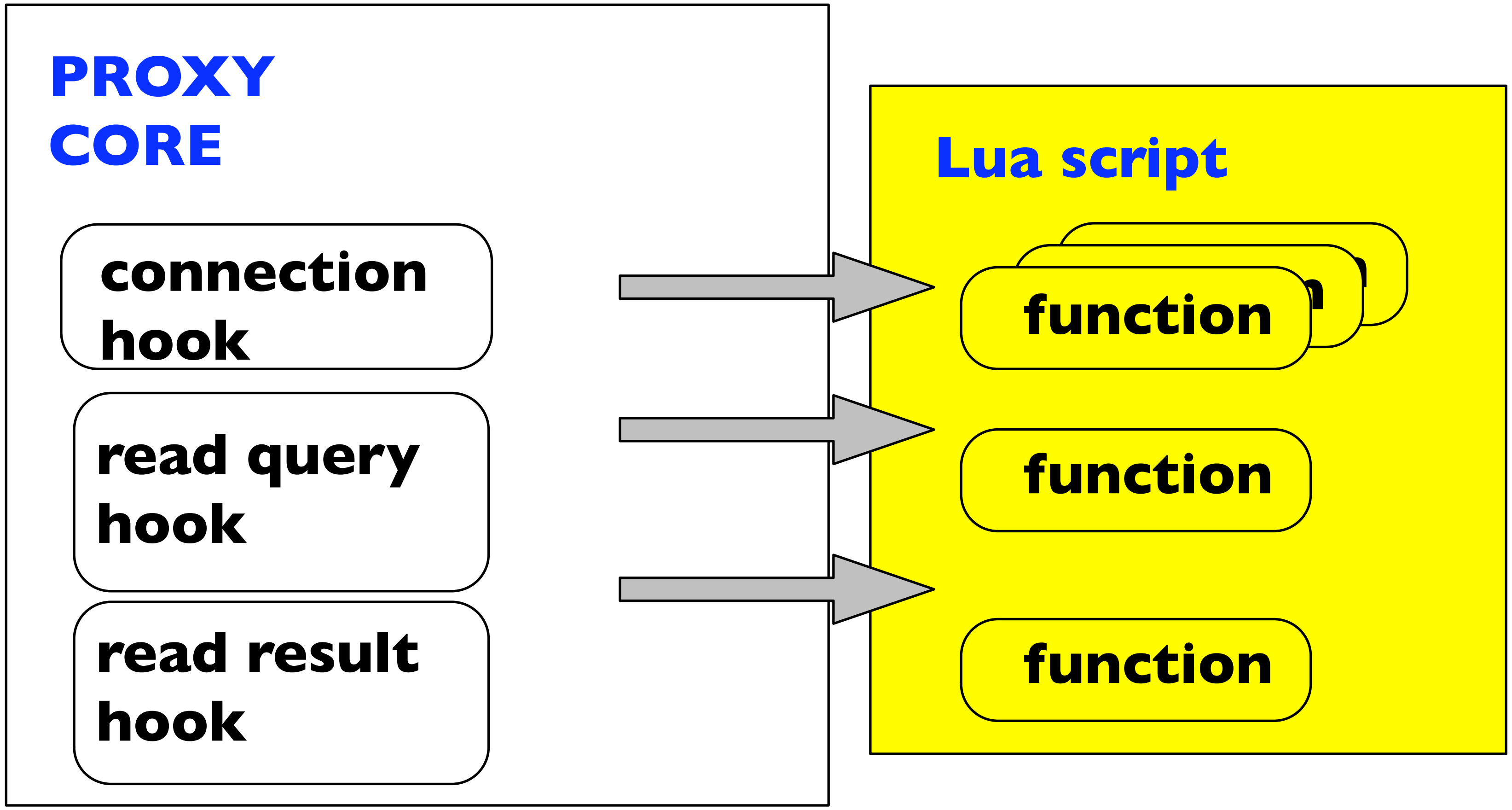
what can you do with MySQL Proxy

Let us show you ...

basic principles



basic principles



Lua

??

Why not ...

Perl ?
PHP?
Javascript?
[whatever]?

Lua

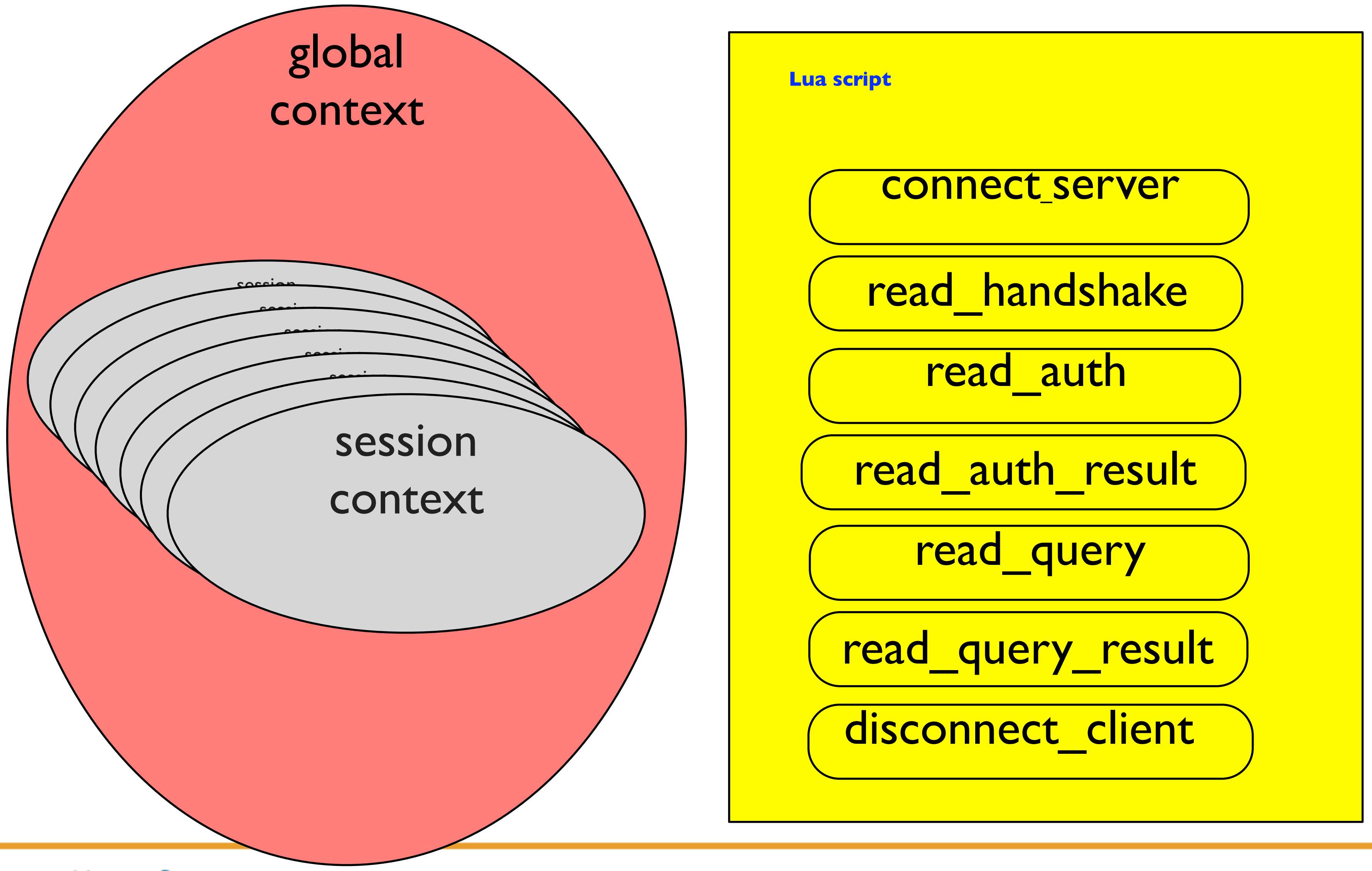
- SMALL (< 200 KB)
- DESIGNED for EMBEDDED systems
- Widely used (lighttpd)

lighttpd, like MySQL Proxy, was created by Jan Kneschke

Lua

Very popular among game writers

Proxy - Lua overview



Using Lua Files

```
/usr/local/sbin/mysql-proxy \
  --proxy-lua-script=/path/name.lua
```

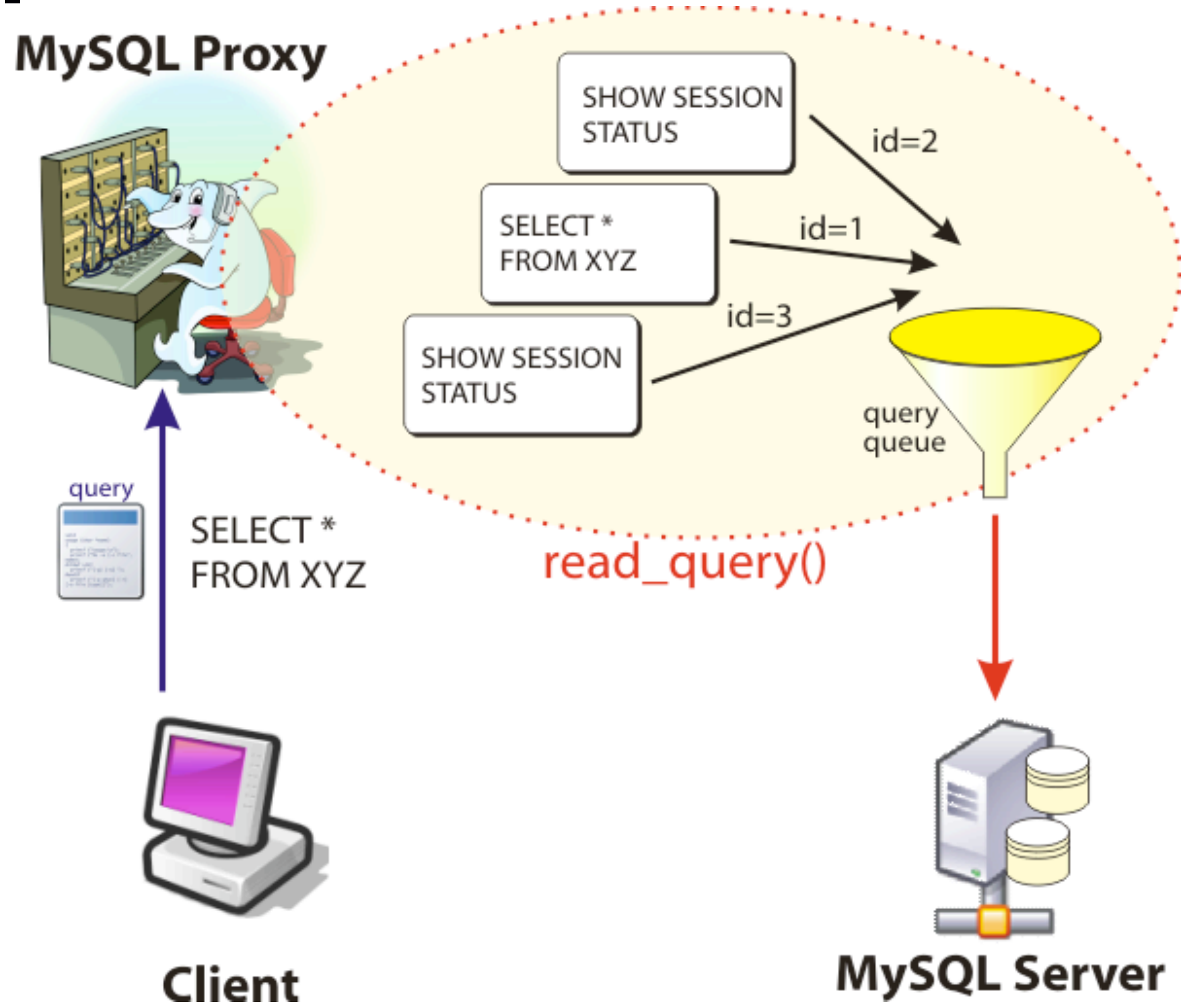
IMPORTANT!
THE SCRIPT DOES NOT START UNTIL THE FIRST CLIENT CONNECTION

intercepting

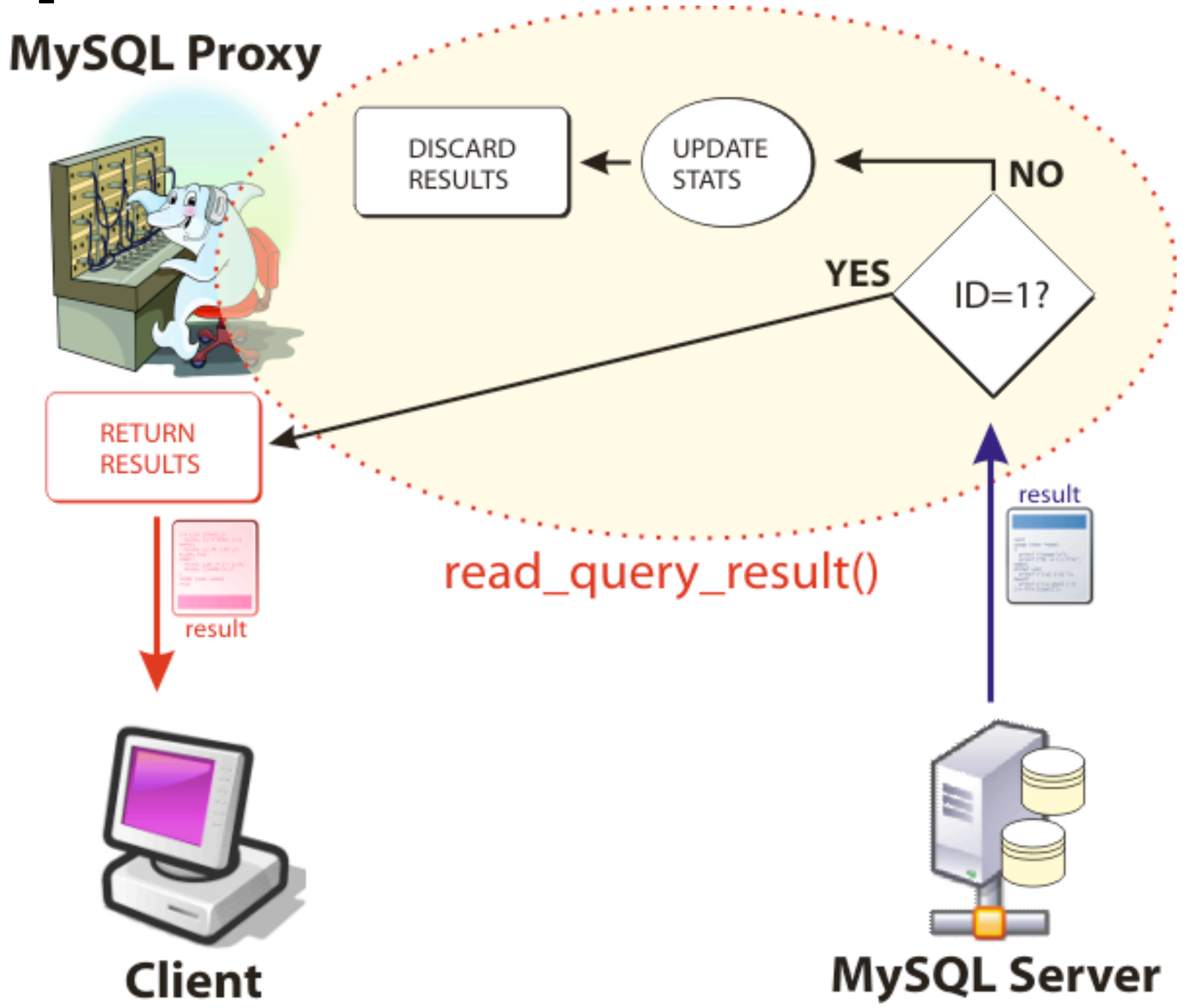
```
function read_query(packet)
 if packet:byte() == proxy.COM_QUERY
   then
   local query = packet:sub(2)
   print("Hello world! Seen query: "
        .. query )
 end
end
```

Presented by

injecting queries



injecting queries



injecting queries

```
function read_query(packet)
  -- ...
  proxy.queries:append(2, query1 )
  proxy.queries:append(1, packet )
  proxy.queries:append(3, query2 )

  return  proxy.PROXY_SEND_QUERY

end
```

injecting queries

```
function read_query_result(inj)

  if res.id == 1 then
    return -- default result
  else
    -- do something
    return  proxy.PROXY_IGNORE_RESULT
end
```

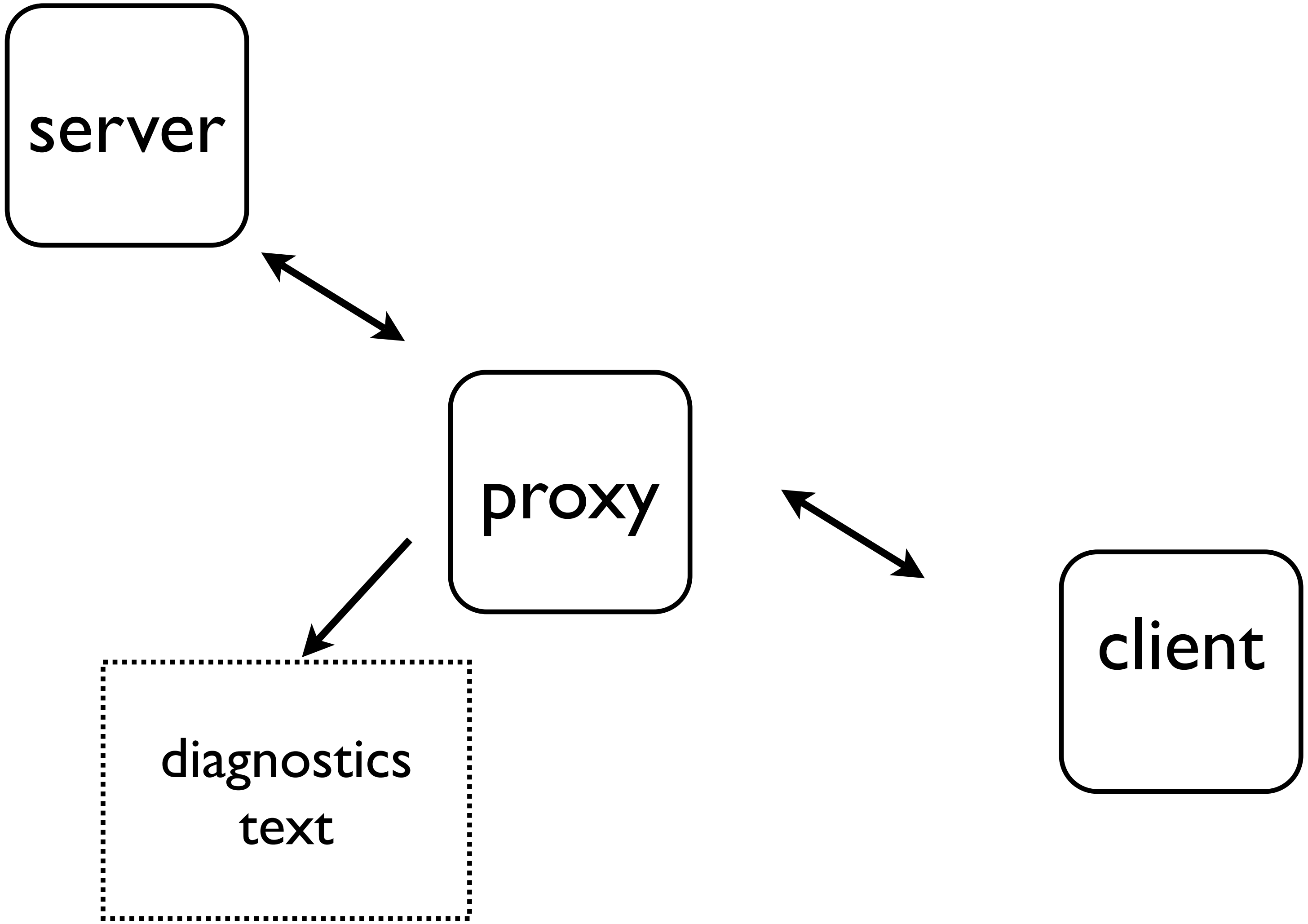
working with results

- return the original result
- return a fake result
- return an error
- alter the original result
- return something different (affected/retrieved)

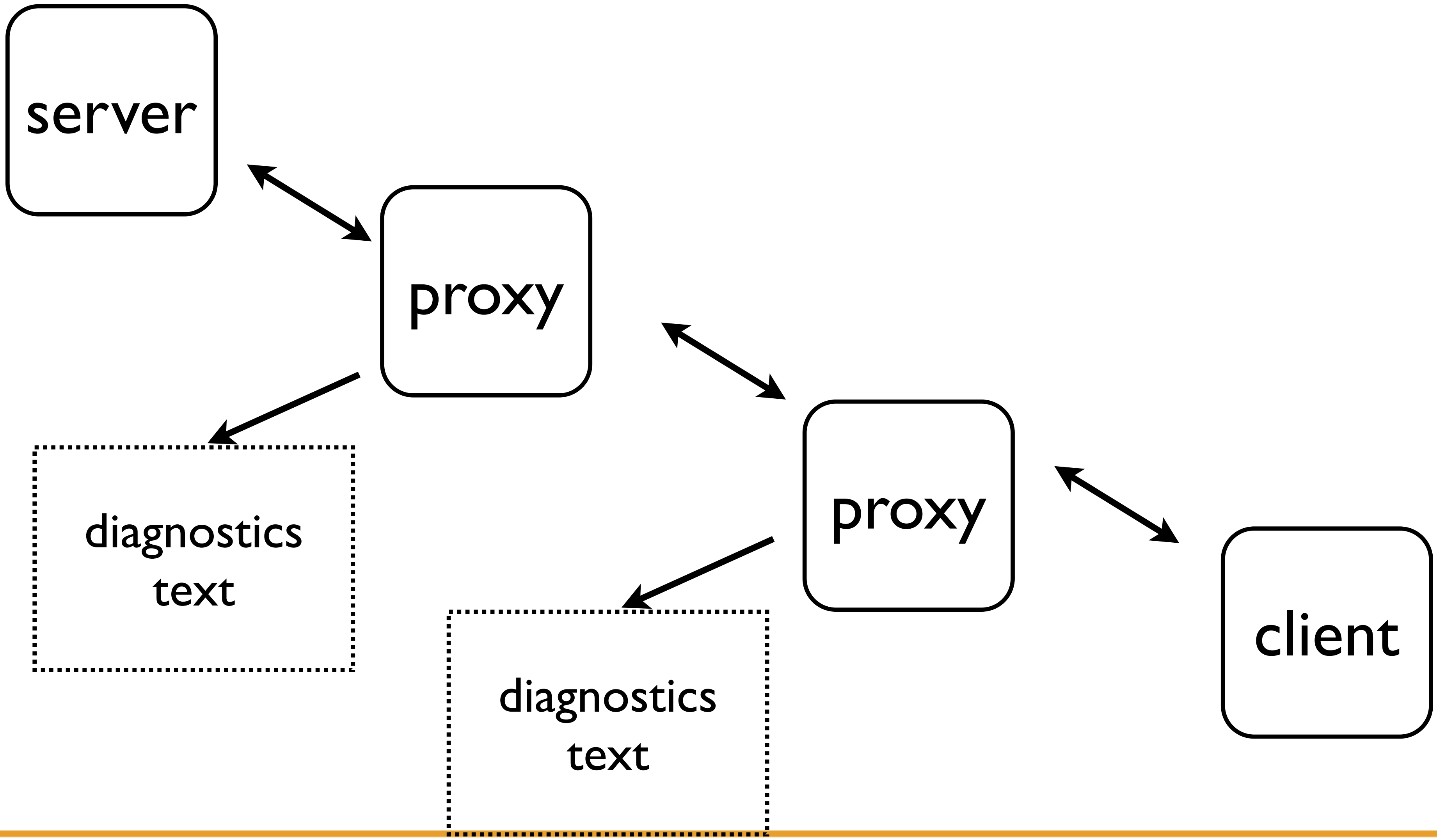
debugging

- Put a Proxy in between
- use a sensible script to see what's going on (e.g. tutorial-packets.lua or tutorial-states.lua)

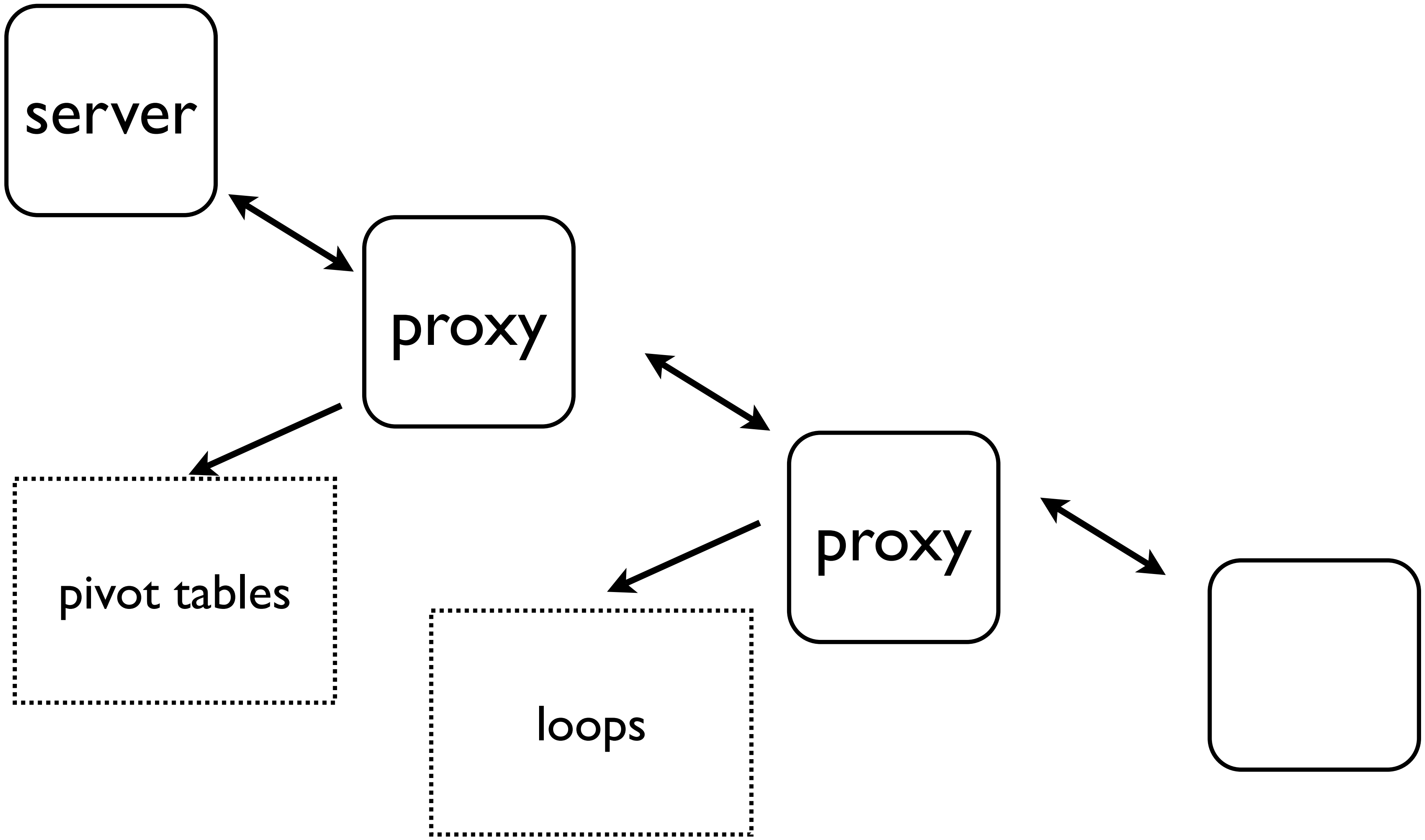
debugging



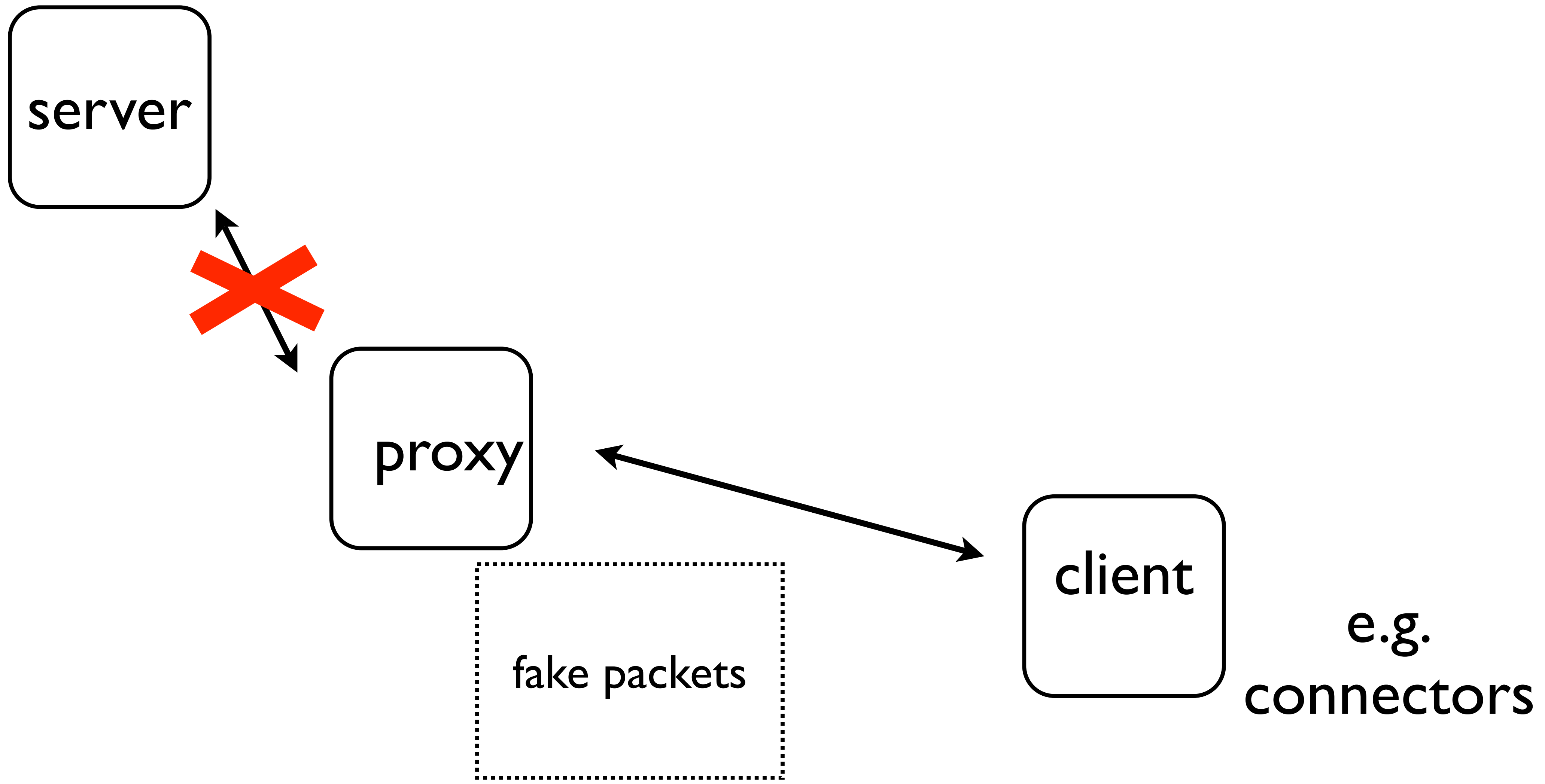
debugging scripts



chained proxy: double features



testing



Lua basics

Lua types

- nil
- number
- string
- table
- function
- userdata

```
a = nil
b = 1
c = 'abc'
t = { a,b,c }
f = print
u = some_C_struct
```

Lua comments

```lua
-- simple comment

print(1)
--[[
  print(2)
  print('hello')
--]]
print(3)
```

Lua comments

```
-- simple comment

--[=[
  print(1)
  --[[
   print(2)
   print('hello')
  --]]
  print(3)
--]=]
```

Presented by

Numbers and strings

- nil = no value (false)
- number = floating point
- '5' =/= 5

```
a = nil

b = 5; c = '5'
print (b == c)
false
print (b == c +0)
true
```

Numbers and strings

- conversion on demand

```
a = 5 ; b = '5'

print(type(a), type(b))
number  string
print(type(b+0))
number
print(type(a .. ""))
string
```

Numbers and strings

- conversion on demand

```
a = 5 ; b = '5'

print(type(tostring(a)))
string
print(type(tonumber(b)))
number
```

strings

- smart quoting

```
a = 'Hello'
b = "World"
c = "Can't"
d = [[Don't say "Hello"]]
e = [=["d'oh" [[braces]]!]=]
```

tables

- associative arrays
- can be used as arrays
- can create complex structures

```
t1 = {10, 20, 30 }
t2 = {
   a = 'abc',
   b = 2,
   c = { 3, 4}
}
```

functions

- can be assigned to variables
- new functions can override existing ones

```
function f (x)
  print(x)
end
g = f
g(10)
```

userdata

- containers to exchange data between Lua and host language
- can have "tag methods"

statements

- normal assignments

```
a = 3
b = get_num() -- func return
```

- multiple assignments

```
a,b = 3,2
```

statements

- multiple return values

```
  function x ()
    return 1, 'OK'
  end
a, b, c = x()
  -- a = 1, b = 'OK', c = nil
```

statement blocks

- if

```
if condition then
   statements
end
```

- while

```
while condition do
   statements
end
```

statement blocks

- for

```
for var = 1, 10 [,step] do
    statements
end
```

- for

```
for n,v in pairs(table_var) do
    statements
end
```

sample function

- read_query

```
function read_query(packet)
   if packet:byte() ~=
     proxy.PROXY_COM_QUERY
   then
     return
   end
   local query = packet:sub(2)
   print('received ' .. query)
end
```

some details

```
== equals
~= not equal

string.byte(packet)
packet:byte()

string.sub(packet,2)
packet:sub(2)

'abc' .. '123'  == 'abc123'
```

using tables

```
t = {}
t[1] = 'a' --First element 1, ! 0
t[2] = 'b'
table.insert(t, 'c')
-- or t[ #t +1 ] = 'c'

t = {'a', 'b', 'c' }
t = {1 = 'a', 2 = 'b', 3 = 'c'}
print (t[2])
b
```

using tables

```
sizes = {}
sizes['john'] = 'XL'
sizes['paul'] = 'M'
sizes['fred'] = 'L'

sizes = {
  john = 'XL',
  paul = 'M',
  fred = 'L',
}
```

using tables

```
sizes = {
   john = 'XL',
   paul = 'M',
   fred = 'L',
}
print(sizes['john'])
XL
print(sizes.paul)
M
```

using tables

```
for i, v in ipairs(t) do
  print (i ..' -> ' .. v)
end

for name,size in pairs(sizes) do
  print(name .. ' ' ..
        wears .. ' ' ..
        size)
end
```

WATCH OUT!

```
/* C / C++ */
int a = 0;
printf("%s\n",
   a ? "true" : "false");
```

```
false
```

```
-- Lua
a = 0
print ( a and "true" or "false")
```

```
true
```

Presented by MySQL Conference & Expo O'REILLY

WATCH OUT!

```
-- Lua
a = false
print ( a and "true" or "false")
false

a = nil
print ( a and "true" or "false")
false
```

Presented by

Finding text

```
query = 'SELECT id FROM t1'

local cmd, column =
   query:match("(SELECT)%s+(%w+)")

if cmd then
   -- do something with query
end
```

finding text

- Regular expressions
- similar to Perl/PHP, but simpler

  - `%` instead of `\`
  - `(captures)`
  - `[character classes]`
  - `^ $ + - ? *`
  - `no alternation (a|b)`
  - `no modifiers /i`

finding text (Proxy way)

```
local tk =
  require('proxy.tokenizer')

local tokens = tk.tokenize(query)

if tokens[1].token_name ==
   'TK_SQL_SELECT'  then
  -- do something with query
end
```

Presented by

finding text (Proxy way)

```
-- each token is a table

token = {
   token_name = 'TK_SQL_SELECT',
   text       = 'select',
   token_id   = 204
}
```

I/O

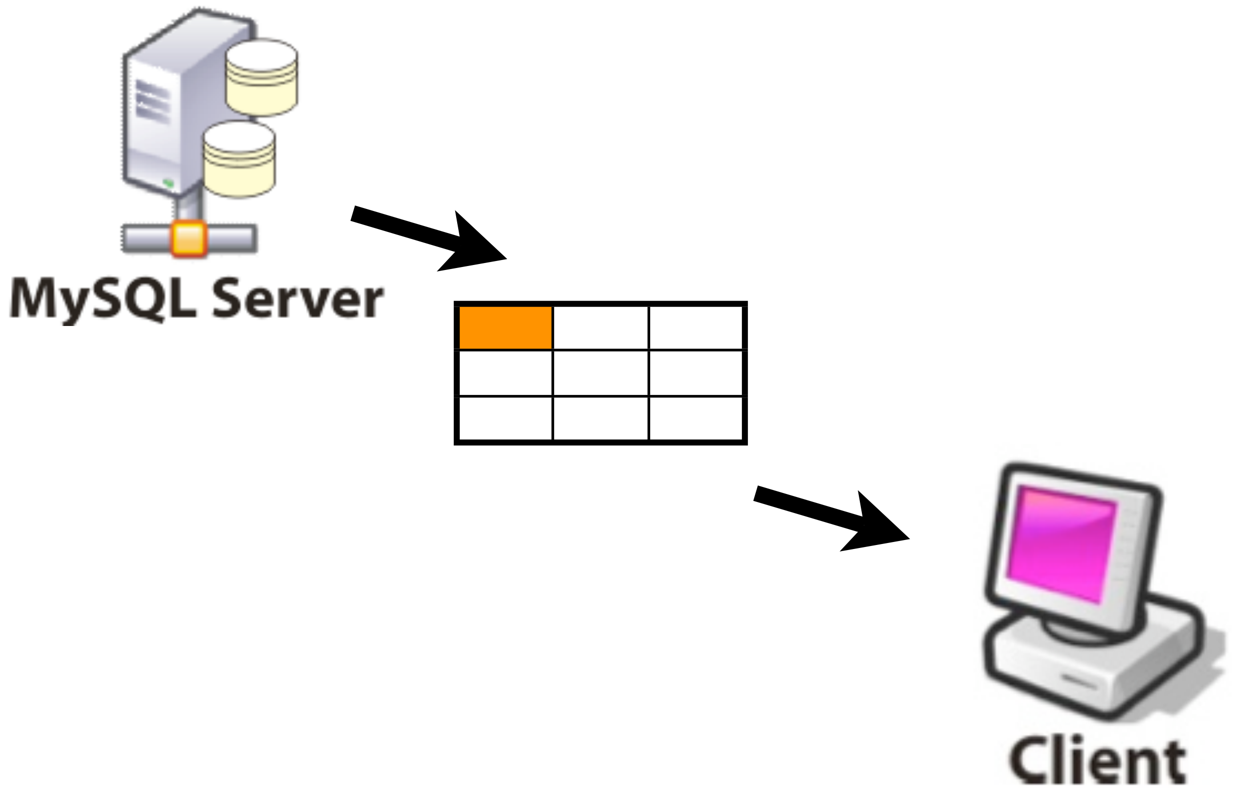
```
-- files are objects

local fname = '/tmp/test.txt'
assert(fh = io.open(fname,'r'))

for line in fh:lines() do
  print(line)
end
fh:close()
```

I/O

```
-- files are objects

local fname = '/tmp/test.txt'

assert(fh = io.open(fname,'w'))
for x = 1, 100  do
    fh:write('new row ' .. x)
    fh:flush()
end
fh:close()
```

MySQL Proxy recipes

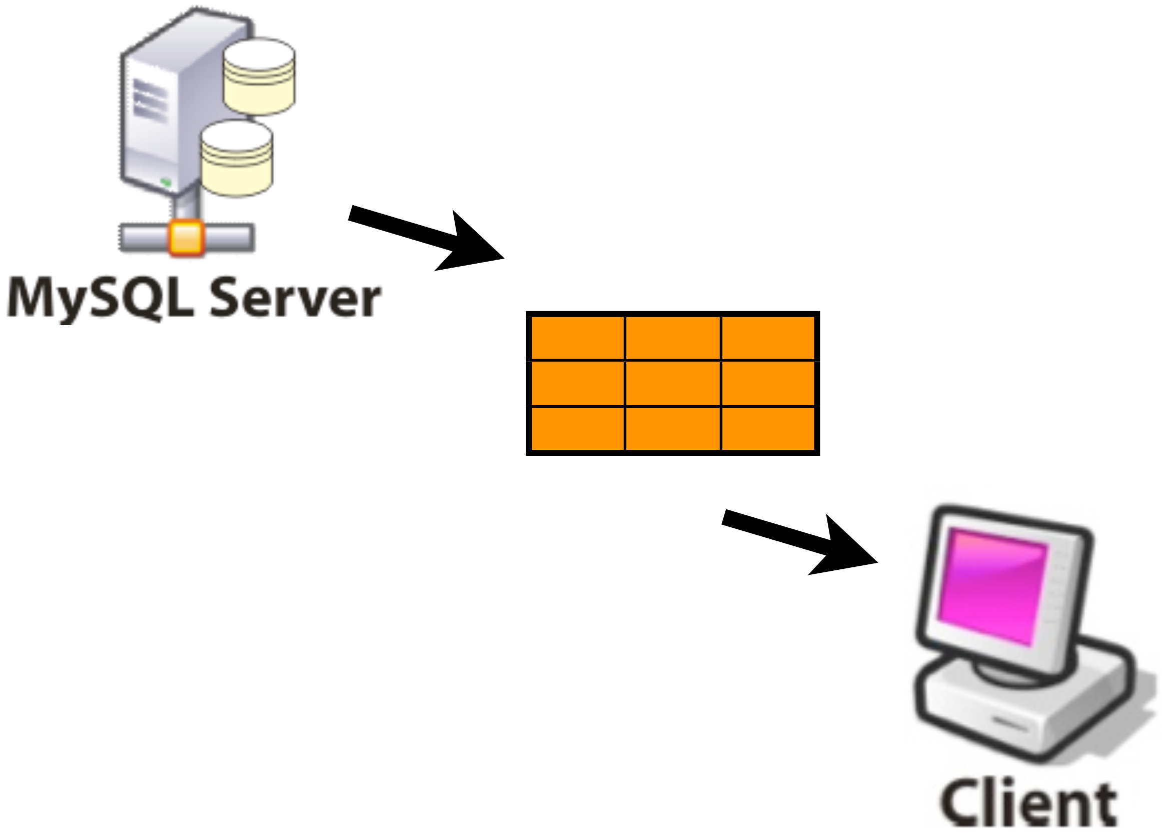
cookbook: returning a simple dataset



cookbook: returning a simple dataset

```
function simple_dataset (header, message)
proxy.response.type = proxy.MYSQLD_PACKET_OK
proxy.response.resultset = {
    fields = {
        {type = proxy.MYSQL_TYPE_STRING, name = header
    },
    rows = {
        { message}
    }
}
return proxy.PROXY_SEND_RESULT
end
```

cookbook: returning a full dataset



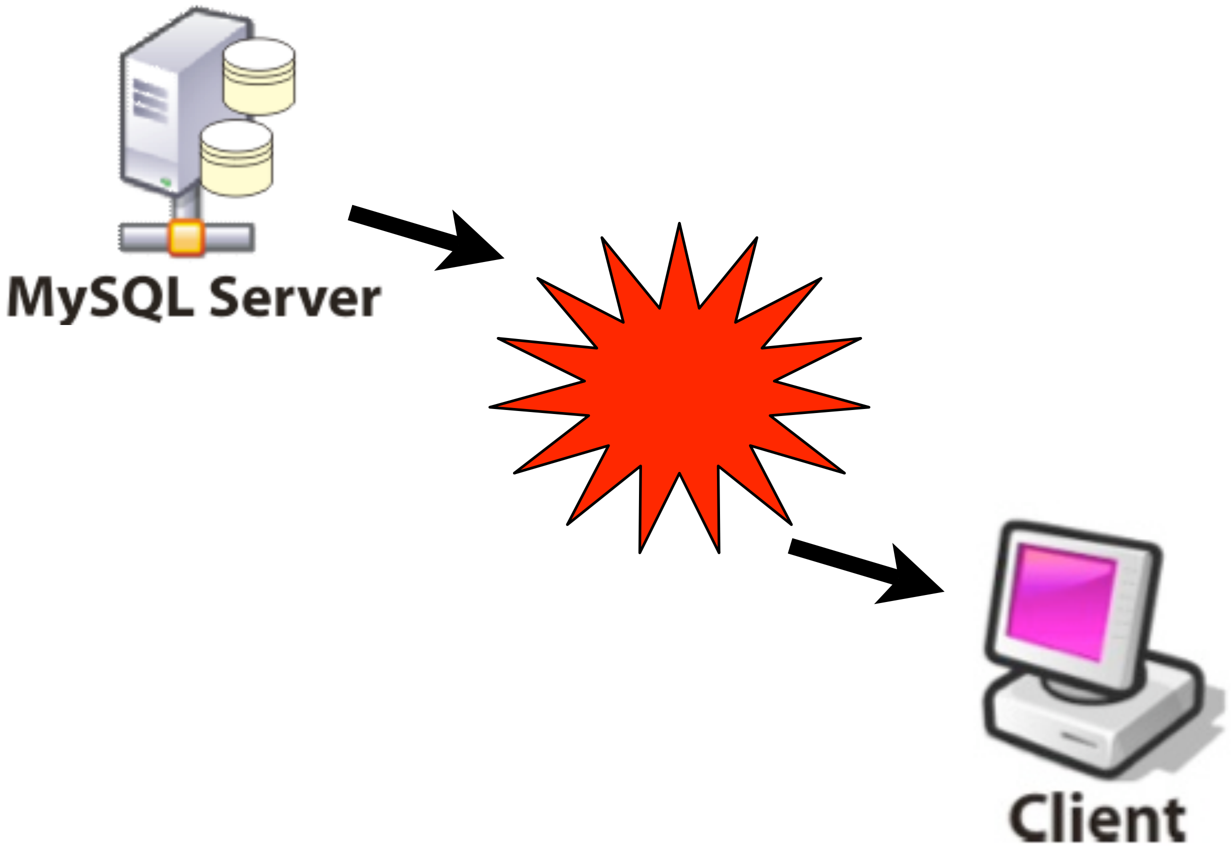
cookbook: returning a full dataset

```
function proxy.global.make_dataset (header, dataset)
  proxy.response.type = proxy.MYSQLD_PACKET_OK

  proxy.response.resultset = {
      fields = {}, rows = {}}
  for i,v in pairs (header) do
      table.insert(
        proxy.response.resultset.fields,
            {type = proxy.MYSQL_TYPE_STRING, name = v
  end
  for i,v in pairs (dataset) do
      table.insert(proxy.response.resultset.rows, v )
  end
  return proxy.PROXY_SEND_RESULT
end
```

cookbook: returning a full dataset

```
return make_dataset(
  {'command', 'description' },         -- the header
  {                                    -- the rows
    {'FOO', 'removes the database'},
    {'BAR', 'drops all tables'},
    {'FOOBAR', 'makes the server explode'},
  }
)
```

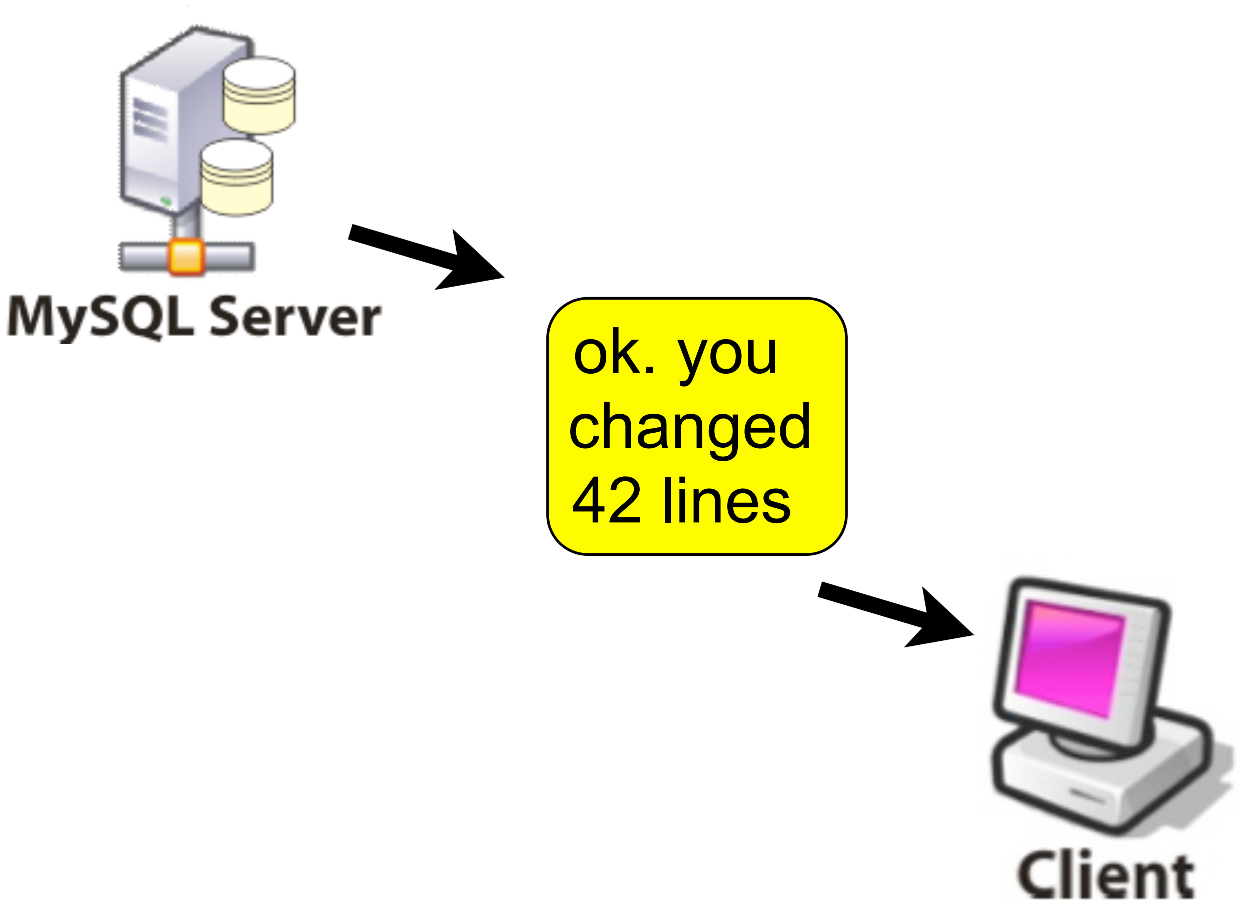
cookbook: returning an error



cookbook: returning an error

```
function error_result (msg, code,state)
  proxy.response = {
      type         = proxy.MYSQLD_PACKET_ERR,
      errmsg       = msg,
      errcode      = code,
      sqlstate     = state,
  }
  return proxy.PROXY_SEND_RESULT
end
```

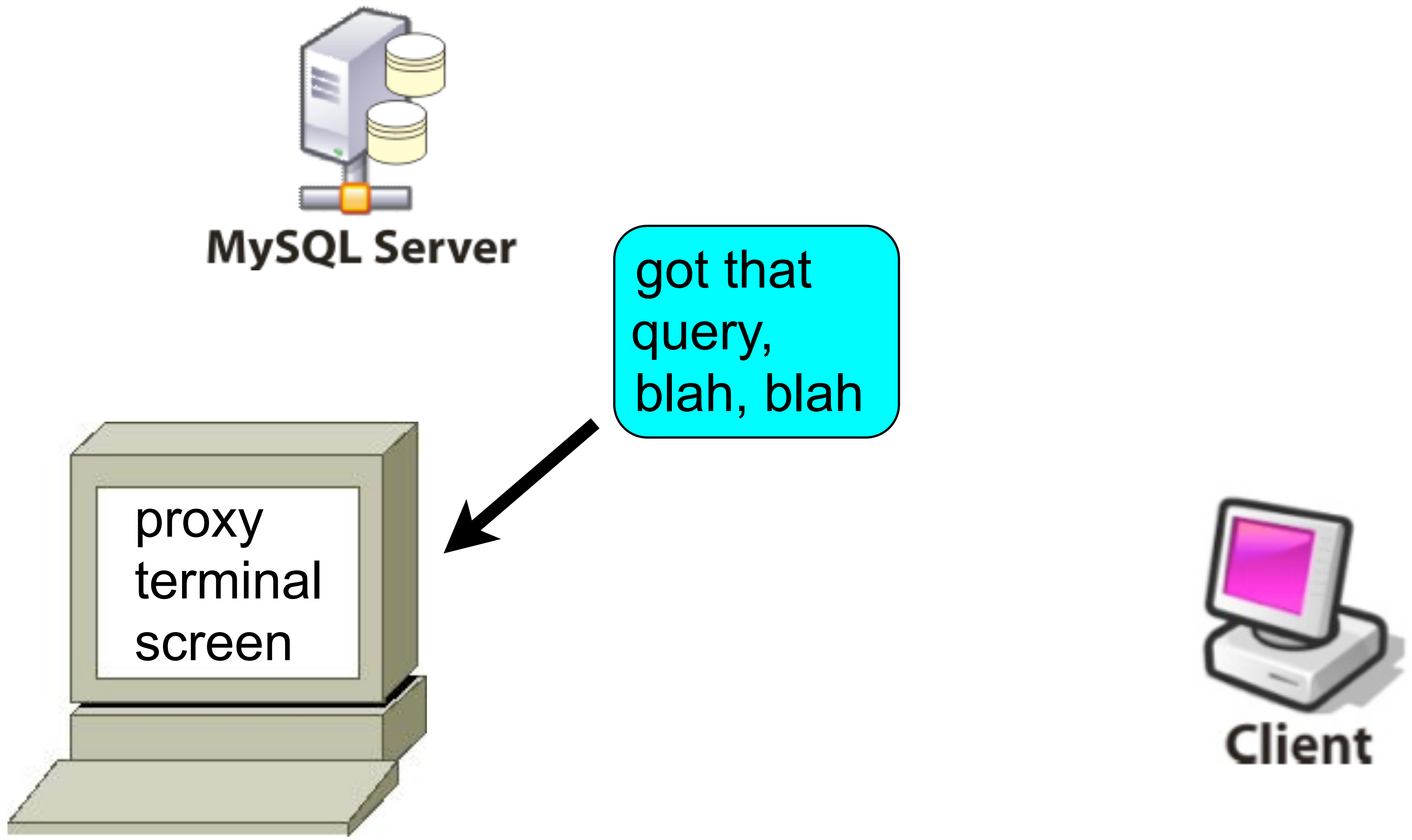
cookbook: returning a non dataset result



cookbook: returning a non dataset result

```
function affected_rows (rows,id)
  proxy.response = {
      type          = proxy.MYSQLD_PACKET_OK,
      affected_rows = rows,
      insert_id     = id,
  }
  return proxy.PROXY_SEND_RESULT
end
```

cookbook: debug messages



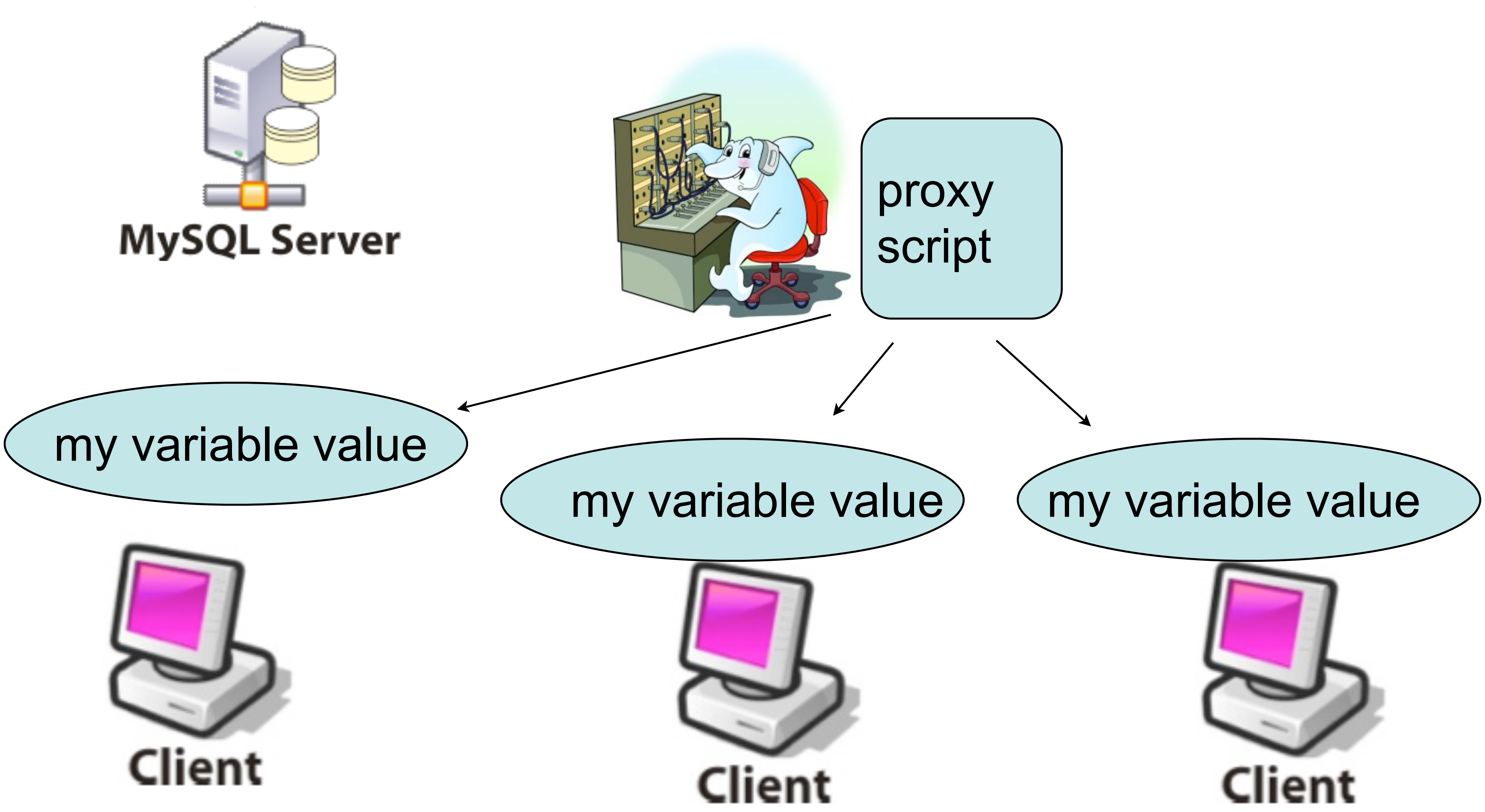
cookbook: debug messages

```
local DEBUG = os.getenv('DEBUG') or 0
DEBUG = DEBUG + 0

function read_query (packet )
 if packet:byte() ~= proxy.COM_QUERY then return end
 print_debug(packet:sub(2),1)
 print_debug('inside read_query', 2)
end

function print_debug(msg, level)
 level = level or 1
 if DEBUG >= level then
     print (msg)
 end
end
```

cookbook: verbose level at run time

```
local DEBUG = os.getenv('DEBUG') or 0
DEBUG = DEBUG + 0

function read_query (packet )
 if packet:byte() ~= proxy.COM_QUERY then return end
 local vlevel=query:match('^VERBOSE=(%d)$')
 if vlevel then
    DEBUG = vlevel+0
    return simple_dataset('verbose',vlevel)
 end
end
```

cookbook: keep info inside a session

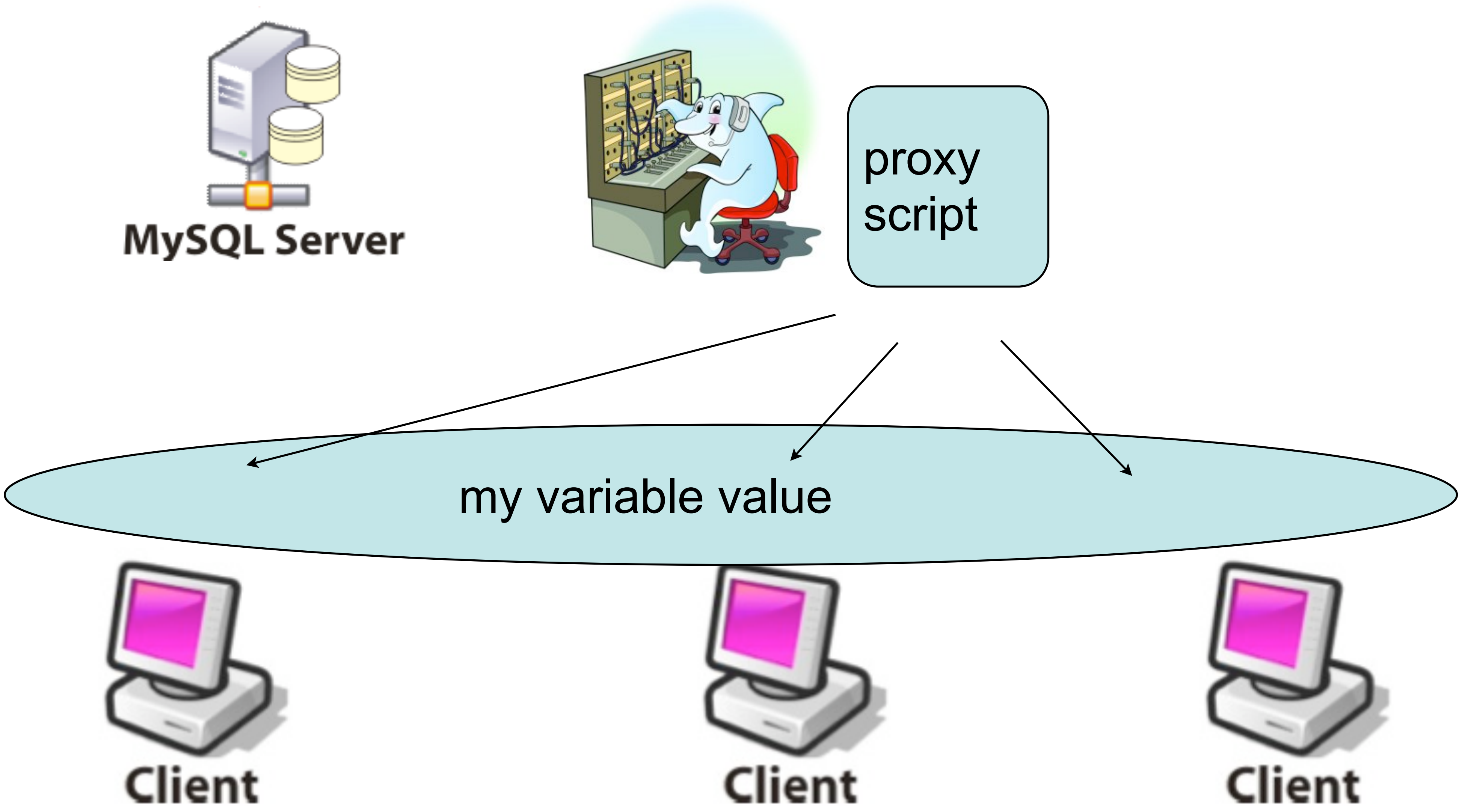


cookbook: keep info inside a session

- nothing to do :)
- Proxy scripts have session scope by default

```
local tot_q = 0

function read_query (packet )
  if packet:byte() ~= proxy.COM_QUERY then return end
  tot_q = tot_q + 1
  print('queries ' .. tot_q)
end
```

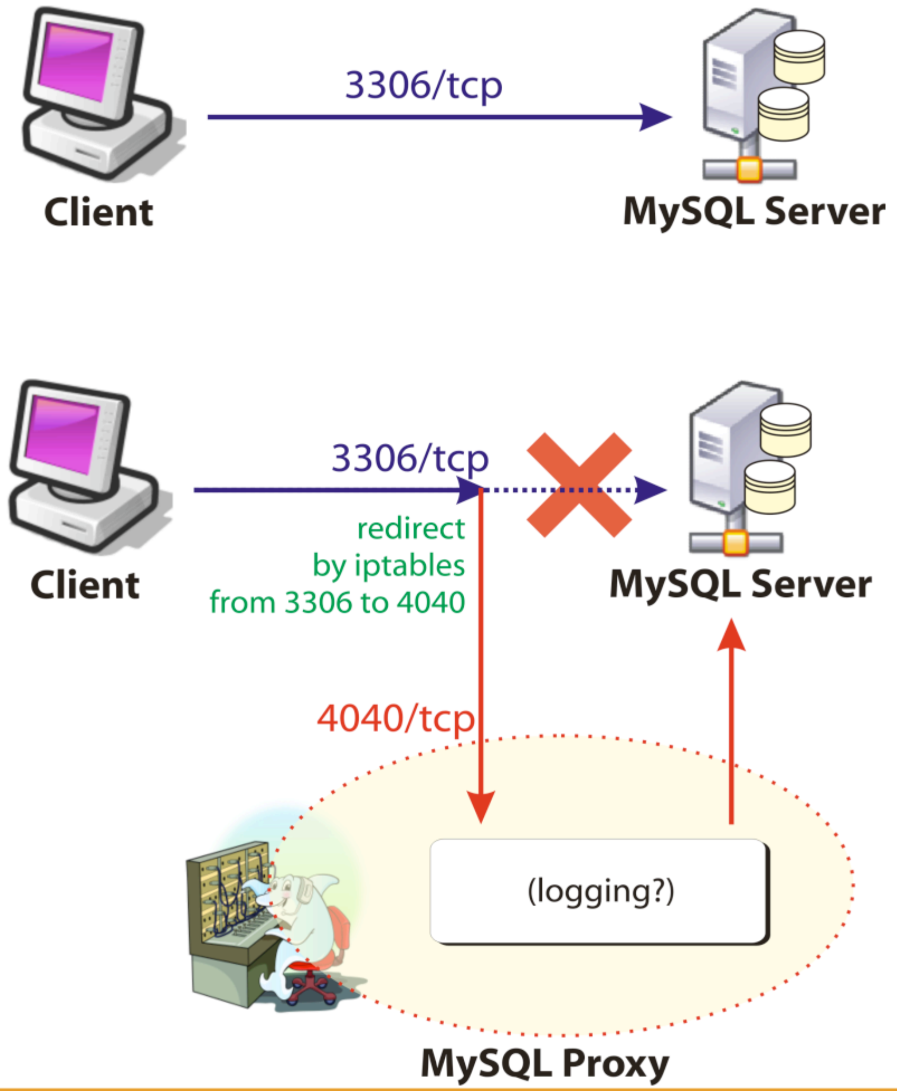
cookbook: share info among sessions



cookbook: share info among sessions

```
proxy.global.tot_q = proxy.global.tot_q or 0

function read_query (packet )
  if packet:byte() ~= proxy.COM_QUERY then return end
  proxy.global.tot_q = proxy.global.tot_q + 1
  print('queries ' .. proxy.global.tot_q)
end
```

cookbook: rerouting traffic



cookbook: rerouting traffic

```
(1) do

sudo iptables -t nat \
  -I PREROUTING \
  -s ! 127.0.0.1 -p tcp \
  --dport 3306 -j \
  REDIRECT --to-ports 4040
```

cookbook: rerouting traffic

```
(1) undo

sudo iptables -t nat \
   -D PREROUTING \
   -s ! 127.0.0.1 -p tcp \
   --dport 3306 -j \
   REDIRECT --to-ports 4040
```

Examples

`http://datacharmer.org/tutorial_uc2008`

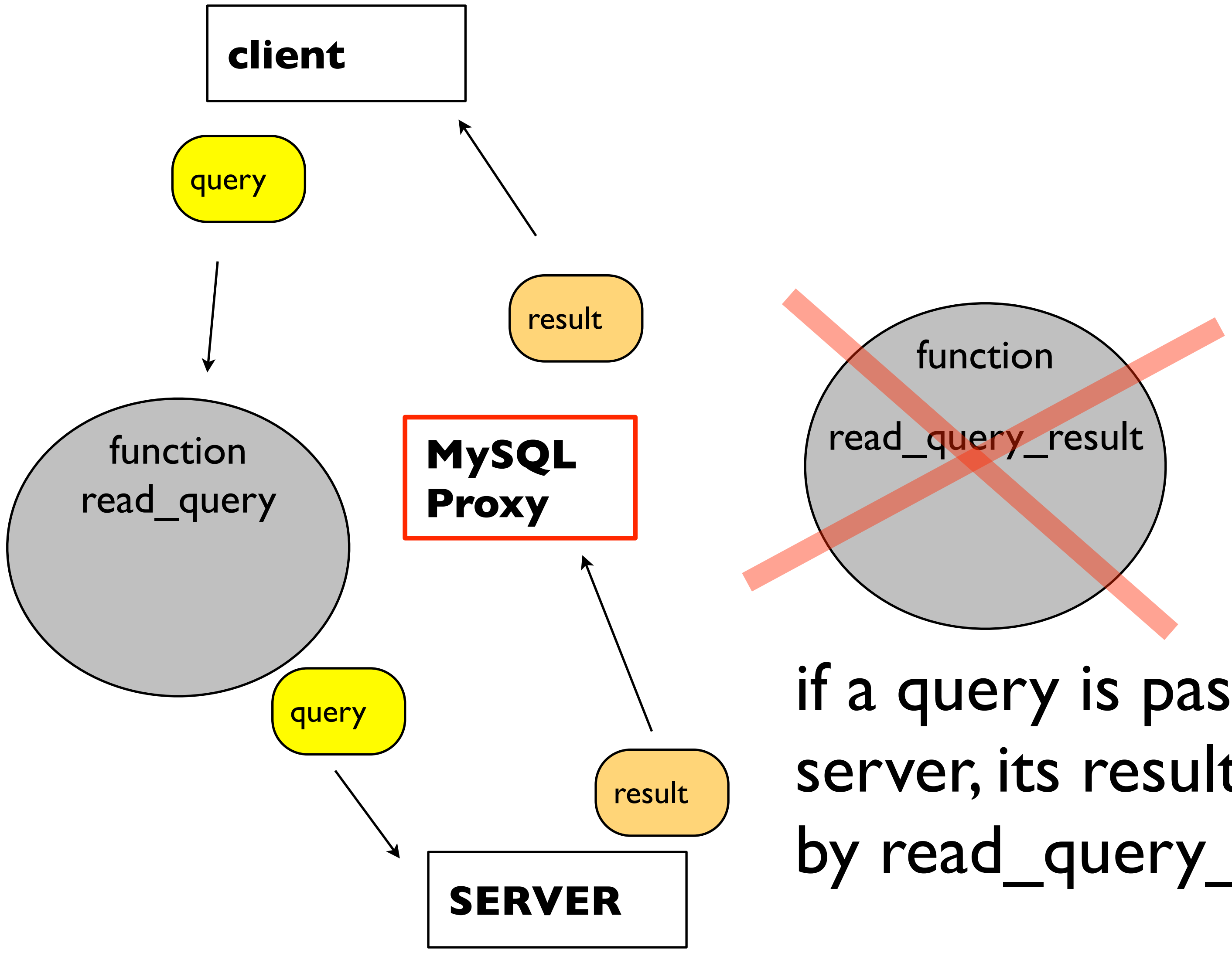
- all hooks
- session bandwidth
- user bandwidth
- blocking commands

all_hooks.lua

source: 010_all-hooks.lua

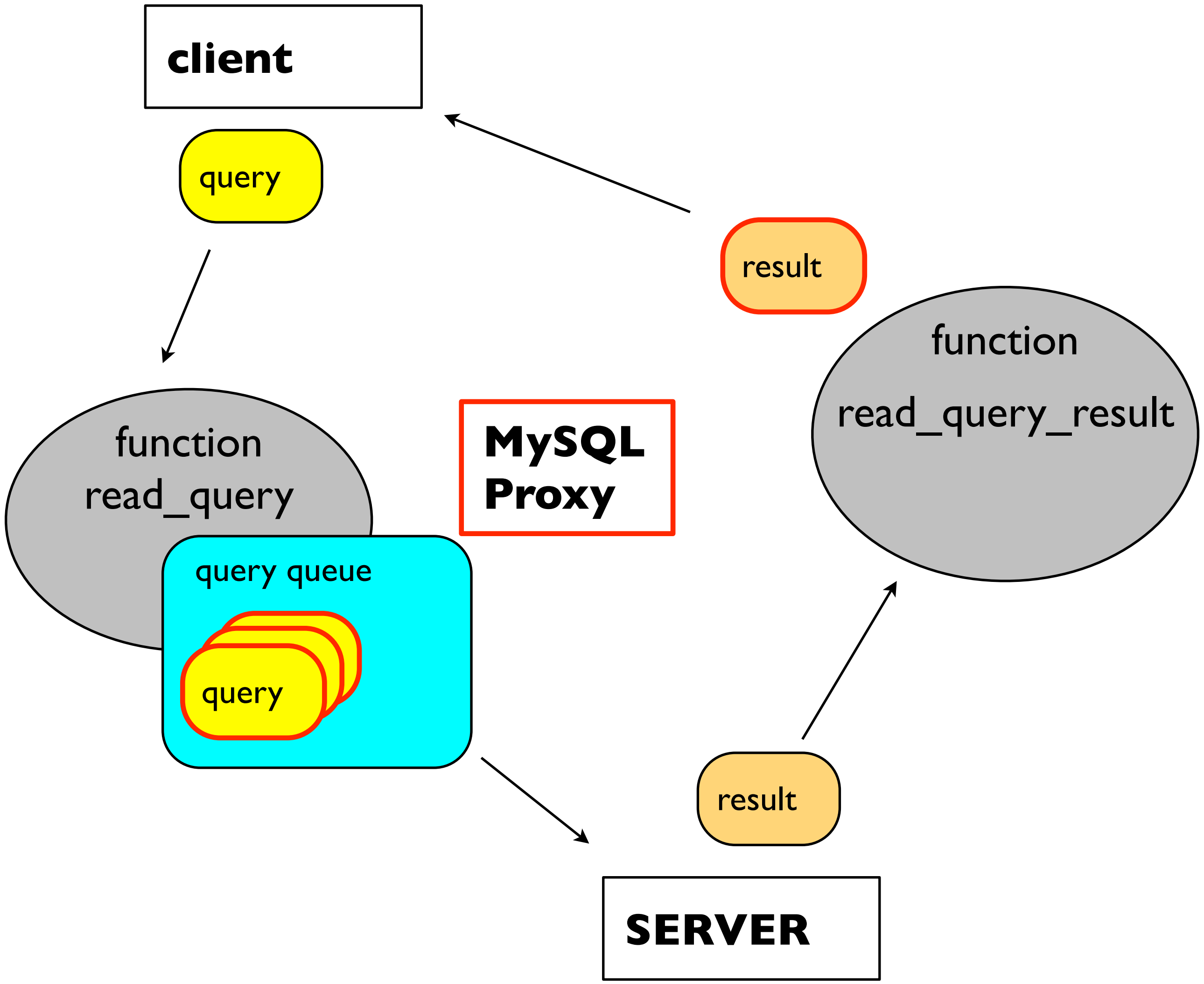
```
function read_query (packet)
    print_access('inside read_query \t' .. packet:sub
    proxy.queries:append(1, packet)
    return proxy.PROXY_SEND_QUERY
end

function read_query_result (inj)
    print_access('inside read_query_result \t' ..
inj.query)
end
```

read_query and read_query_result



if a query is passed directly to the server, its result is **NOT** evaluated by read_query_result

read_query and read_query_result



only if a query is added to the **query queue**, its result is evaluated by read_query_result

all_hooks.lua

source: 010_all-hooks.lua

```
sample output
/usr/local/sbin/mysql-proxy --proxy-lua-script=all-
hooks.lua
 1 inside connect_server
 2 inside read_handshake
 3 inside read_auth
 4 inside read_auth_result
 5 inside read_query
 6 inside read_query_result
 7 inside read_query
 8 inside disconnect_client
```

more examples

- live

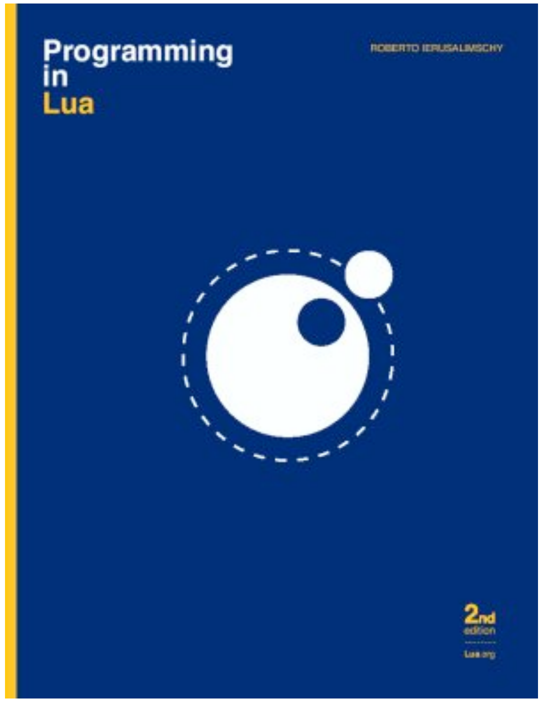
read more

http://www.lua.org/docs.html

online Lua documentation

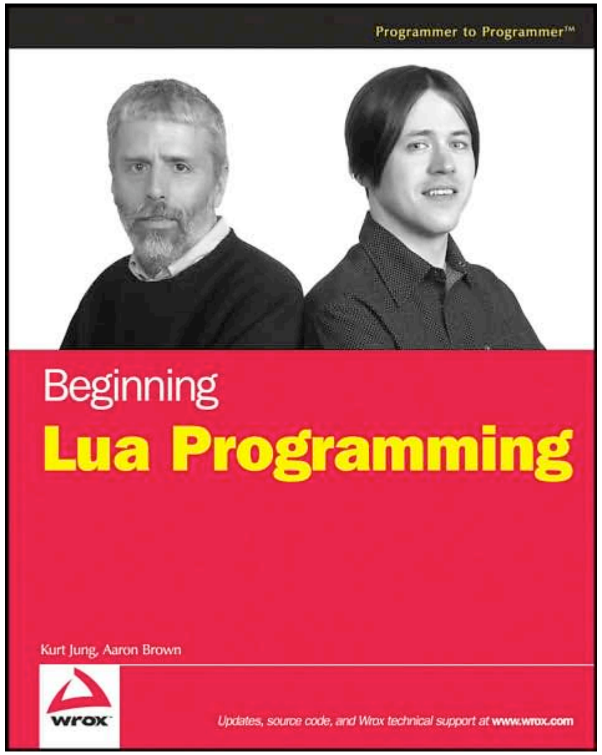
read more

http://www.inf.puc-rio.br/~roberto/pil2/



read more

http://www.wrox.com/WileyCDA/WroxTitle/productCd-0470069171.html


Programmer to Programmer™
Beginning
Lua Programming
Kurt Jung, Aaron Brown
wrox
Updates, source code, and Wrox technical support at www.wrox.com

Q&A