A new Lua bridge to a (rocksolid, lightning fast, secure) Database engine

Alain Descamps
Brocade – Library & Archive Software
Anet – University of Antwerp Library

Who is Alain ?
> 38 years M
versus
> 3.8 hours Lua

The database engine: M(umps):

- noSQL
- ISO standard
- Open Source version(s) available

Used, but hidden:

- Government:
  - UAntwerpen Library,
  - Federal Police (Belgium),..
- Healthcare:
  - Orville Hospital (N. California),
  - Hakeem heathcare system (Jordan),..
- Financial
- Gaia space project (ESA)
- Metro Kiev (Ukraine) (?)
- ...

M(umps)
=

⟺ A
database

+

⟺ A programming language
(the only bridge, so far)

The M database:

Rock solid:
- Transactional processing
- Online backup
- Mirroring
- ...

Lightning fast:
- 1000 fin. transactions/sec barrier
- multi user (R/W) by nature

Secure:
- Unix-alike permission control

University of Antwerp

The M language:

- build for access to M database
- direct, high-speed
- simple & primitive
- durable
- scalable (from Raspberry Pi to Multi-CPU)
- philosophical similarities with Lua

- no coroutines
- no iterators
- no classes
- no batteries included (but we were diligent)
- limited interoperability with outside world

anet.be

(So we started a revolution..)

⟺ M database

\+

⟺ Lua bridge to M
(to boldly go..)

Lua bridge to M:

- https://github.com/orbitalquark/lua-yottadb.git (Open Source)
- based on YottaDB.com (Open Source M)
- developed by Mitchell (USA) (textadept)
- Berwyn Hoyt (NZ)
- sponsored by University Antwerpen

M versus Lua syntax : example : Lua(1)

```
books = {}
books[14502527] = {}
books[14502527]["title"] = {}
books[14502527]["title"]["lg"] = "eng"
books[14502527]["title"]["ti"] = "Programming in Lua"
books[14502527]["edition"] = "4 ed."
```

M versus Lua syntax : example: M (1)

```
s books(14502527,"title","lg")="eng"
s books(14502527,"title","ti")="Programming in Lua"
s books(14502527,"edition")="4 ed."
```

(s = set)

M versus Lua syntax : example: M(2)

```
s ^books(14502527,"title","lg")="eng"
s ^books(14502527,"title","ti")="Programming in Lua"
s ^books(14502527,"edition")="4 ed."
```

= Global variable:

- Written on disk
- Common for all M processes

M variables are always sorted:

^books(14502527,"edition")="4 ed."

^books(14502527,"title","lg")="eng"

^books(14502527,"title","ti")="Programming in Lua"

M global variables are (very) scalable:

- Ex.: ^BLNL("trl","UA",115399485)

M versus Lua syntax : example : Lua wrapper(2)

```
m = require('yottadb')
m.set("^books", {14502527, "title", "lg"}, "eng")
m.set("^books", {14502527, "title", "ti"}, "Programming in Lua")
m.set("^books", {14502527, "edition"}, "4 ed.")
```

M versus Lua syntax : example : Lua wrapper(2)

```
m = require('yottadb')
m.set("^books", {14502527, "title", "lg"}, "eng")
m.set("^books", {14502527, "title", "ti"}, "Programming in Lua")
m.set("^books", {14502527, "edition"}, "4 ed.")
```

+ 16 available functions (now):

- set
- lock
- delete node
- .. (all you need is there)

But..

M↔Lua : future plans:

- Integration with M more Lualike ?
- M global variables act as Lua tables (M/Lua wrapper)
- Accessing Lua from within M (Berwyn Hoyt) (M/Lua bridge)
- proof-of-concept/deployment in Brocade (Library Software system)

M↔Lua : future plans:

- Integration with M more Lualike ?

Proposal (Berwyn Hoyt):

- https://bitbucket.org/berwynhoyt/mlua/src/tasks/proposal.md

Discussion
Questions – Suggestions ?

https://anet.be/

Alain Descamps

alain.descamps@uantwerpen.be