ztsdb, a time-series DBMS for R users

R/Finance 2017
May 19, 2017

Leonardo Silvestri
lsilvestri@ztsdb.org

What is a time-series DBMS?

- time-series: set of values associated with an ordered set of times
- services: insertion, management, storage and retrieval
- optimized for append
- aggregation

Main concepts of ztsdb

- fast, small multi-user noSQL column-store
- query and manipulation language: subset of R
- tight integration with R
- coherent and rich representation of time
- streaming transformations
- seamless connectivity with other ztsdb instances

Query language: ~ R subset

- time-series creation

```
idx <- Sys.time() + as.duration(1:10)
data <- matrix(1:30, 10, 3, dimnames=list(NULL, c("one", "two", "three")))
z <<- zts(idx, data, file="/tmp/my_first_zts")
```

- usual array functionality

```
head(z)
tail(z, 1)
z[1:3, c("one", "two")]
cbind(z, z + z)
```

- temporal types
- escape operator
- pass by reference operator

Integration with R

- ztsdb queries can be written unquoted

```
library(rztsdb)
library(xts)
c1 <- connection(ip="193.143.12.24", port=20001)

c1 ? z
c1 ? head(z)
c1 ? z[1:3, c("one", "two")]
z_local <- (c1 ? z)
```

- seamless data exchange

```
c1 ? Sys.time() - ++Sys.time()
c1 ? rbind(matrix("remote", 2, 2), ++matrix("local", 2, 2))
rbind(matrix("local", 2, 2), c1 ? matrix("remote", 2, 2))
```

- type mapping
  - identical for numeric, character, logical, list
  - zts ↔ xts, time ↔ POSIXct

Time (1)

- POSIX, nanosecond precision, time-zone awareness
- *time, duration, interval, period*

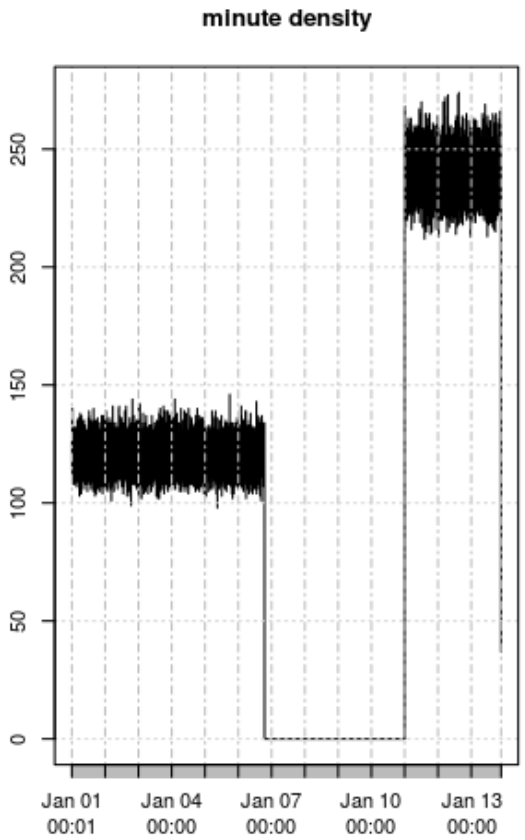
```
start <- as.time("2017-01-01 00:00:00 UTC"); end <- as.time("2017-01-01 00:00:00.001 UTC")

end - start                                  ## [1] 00:00:00.001
ival1 <- as.interval(start, end)
ival2 <- |+2016-01-01 00:00:00 UTC -> 2017-01-01 00:00:00 UTC-|
z[ival2, ]

p <- as.period("1y6m1d/00:01:00.001")  ## [1] 18m1d/00:01:00.001
`+`(start, p, "America/New_York")      ## [1] 2018-07-02 23:01:00.001000000 UTC
`+`(start, p, "UTC")                   ## [1] 2018-07-02 00:01:00.001000000 UTC
```

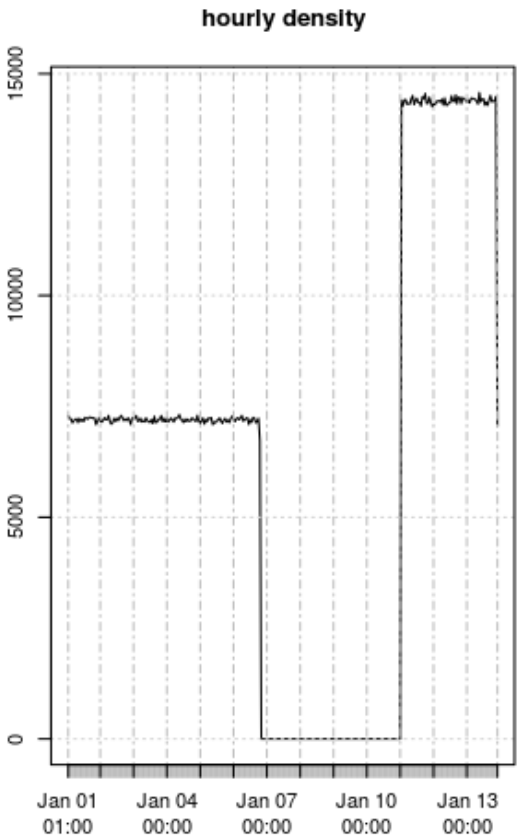
- specialized functions: *align, align.idx, seq*, ...

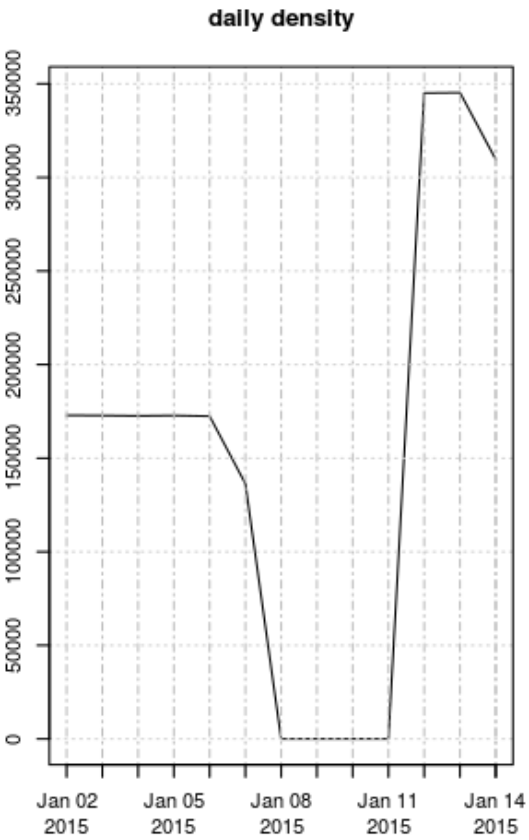
```
minutes <- (c1 ? density(z, as.duration("00:01:00")))
hours   <- (c1 ? density(z, as.duration("01:00:00")))
days <- (c1 ? density(z, as.period("1d"), tz="America/New_York")
```


tsZdb

Time (2)


minute density
0
50
100
150
200
250
Jan 01
00:01
Jan 04
00:00
Jan 07
00:00
Jan 10
00:00
Jan 13
00:00


hourly density
0
5000
10000
15000
Jan 01
01:00
Jan 04
00:00
Jan 07
00:00
Jan 10
00:00
Jan 13
00:00


daily density
0
50000
100000
150000
200000
250000
300000
350000
Jan 02
2015
Jan 05
2015
Jan 08
2015
Jan 11
2015
Jan 14
2015

Streaming transformations

- timer signature

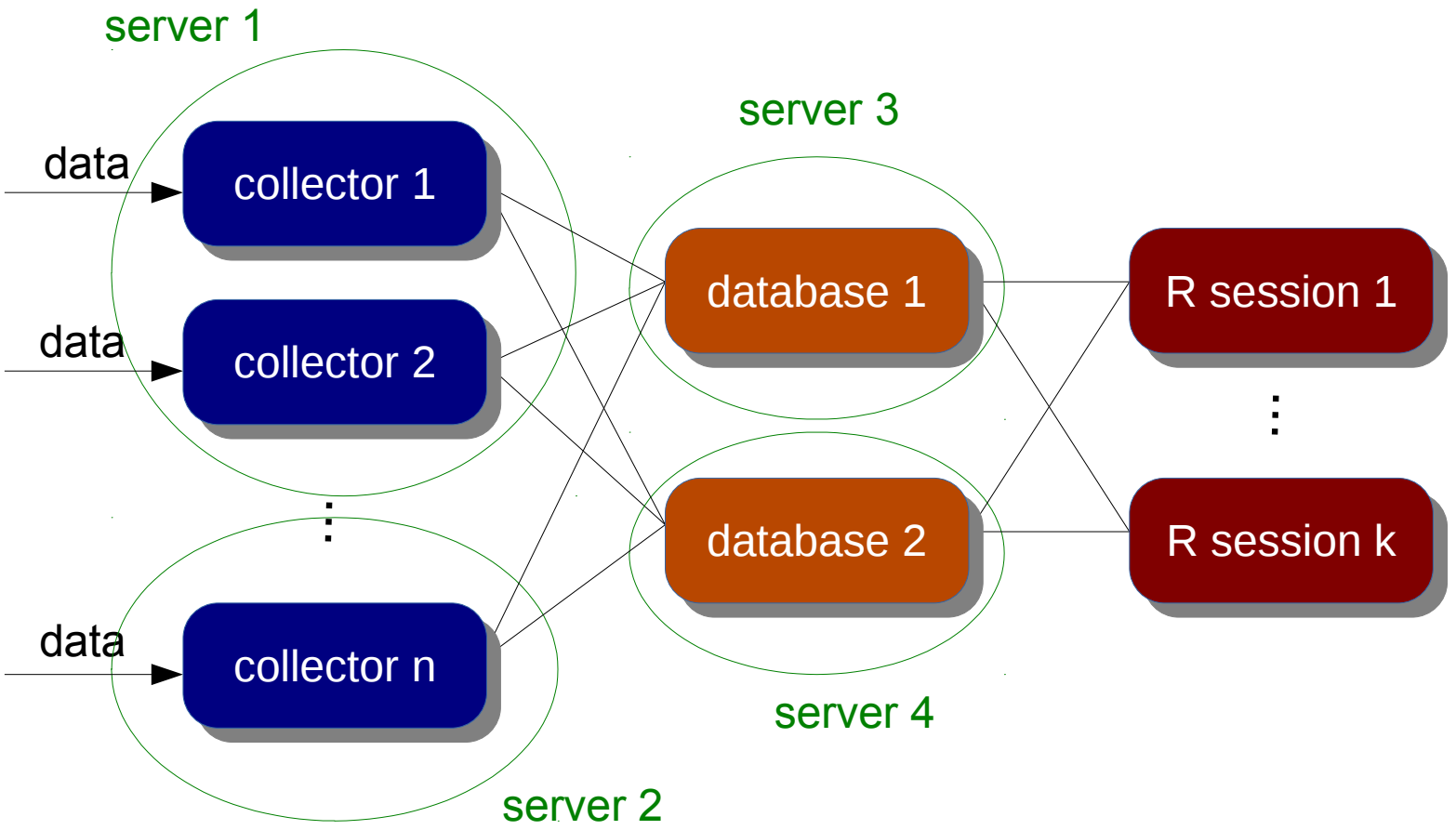
```
timer <- function(duration, loop, once=NULL, loop_max=0)
```

- executes in its own context
  - can create local variables
  - has access to global variables
- example

```
loop = {
  current_minute <- floor(tail(zts.idx(z), 1), "minute")
  if (current_minute >= last_minute + one_minute) {
    last_minute <- current_minute
    m <- align(z, last_minute, -one_minute, method="mean")
    rbind(--mmean, m)
  }
}
```

Connectivity

- data push/pull is possible between any instance
- mechanism for distribution and/or fault tolerance



Info

- Project homepage: http://www.ztsdb.org
- DBMS: https://gitlab.com/lsilvest/ztsdb
- R package: https://gitlab.com/lsilvest/rztsdb
- Docker: https://hub.docker.com/r/lsilvest/ztsdb
- Examples: https://gitlab.com/lsilvest/ztsdb/tree/master/examples