

MUMPS (the language) is  
(nowhere near) Dead;  
Long Live MUMPS  
(the database)!

A Way for VistA to Go Forward



- A mature, high performance, hierarchical key-value NoSQL database whose code base scales up to mission-critical applications like large real-time core-banking and electronic health records, and also scales down to run on platforms like the Raspberry Pi Zero, as well as everything in-between.
- *Rock Solid. Lightning Fast. Secure. Pick any three.*

YottaDB is a registered trademark of YottaDB LLC

- Cities are never designed to become obsolete
- Julius Caesar knew London, Paris, and Rome
  - But he wouldn't recognize any of them today
- Successful cities evolve
  - Those that can't adapt, die, e.g., Copán
  - Continuous evolution inevitably mixes technologies
    - e.g., Fibre-optic networking with medieval sewers

- No planned obsolescence
  - But regular attempts to kill it by starvation
    - [Not unlike laying siege to a city]
- Still recognizable to its original developers
- Adaptation / evolution still in doubt
  - Separating political from technical issues not simple
  - Technology exists to evolve; is the political will there?

# What Defines a City?

- Location and people
- Individuals, buildings, roads, railways, etc. are transient

# What Defines VistA?

- Data about people
- Individuals, interfaces, logic, etc. are transient
  - Even programming languages like M[UMPS]

# What Makes MUMPS Unique?

- Tight binding of database to language
- Other features are powerful, but secondary

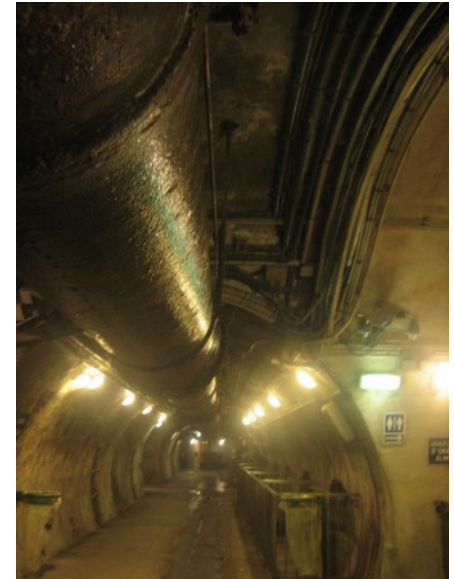
# What Makes MUMPS Problematic?

- Insular community
- Frozen evolution
- Even the biggest vendor won't call it by name
- It's not something the next generation of programmers wants on their resumes
  - And we're too small a community to change that



# How Do Cities Evolve?

- Build on the old and embrace the new
  - With selective, geographically limited redevelopment from time to time
- Coexistence



# How Can VistA Evolve?

- Provide new ways to access and use the data
  - With selective, redevelopment from time to time of limited functional areas
- Coexistence
  - Design new functionality to benefit from old functionality and to allow old functionality to benefit from it

# YottaDB Approach

- Build on what works well
- Accommodate what's new



Public domain from Wikimedia Commons



By GT1976 ICC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>), from Wikimedia Commons

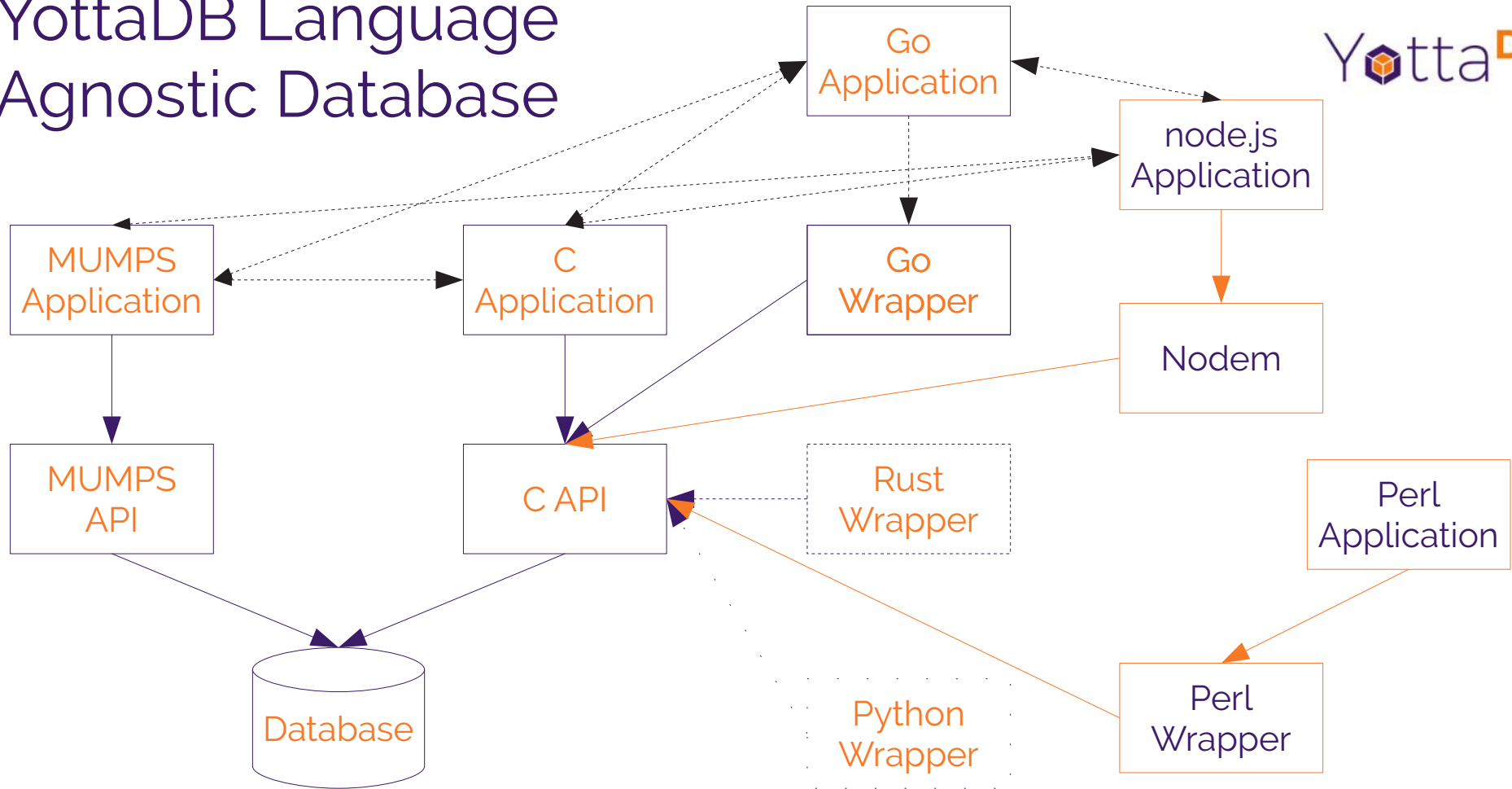
Photos  
are  
almost  
100  
years  
apart

# The YottaDB View



- The diamond is the database
- The language is what it is
  - Like anchovies on pizza, you either love it or hate it
- Solution: language agnostic database
  - Take nothing away from MUMPS, the language
  - Make MUMPS, the database, accessible from other languages

# YottaDB Language Agnostic Database



# Go

- Developed by Google and used internally
  - <https://golang.org>
- Released as free / open source
  - Active user community
  - Growing popularity
- YottaDB's first wrapper
  - <https://yottadb.com/yottadb-go-wrapper/>

## Go co-designer Rob Pike:

- <https://talks.golang.org/2012/splash.article>

... Go's design considerations include *rigorous dependency management*, the adaptability of software architecture as systems grow, and *robustness across the boundaries between components*. ... Go is a *compiled, concurrent, garbage-collected, statically typed* language developed at Google. It is an open source project: Google *imports the public repository* rather than the other way around. .... Go is *efficient, scalable, and productive*.

- Data, Delete, DeleteExcl, Incr, Lock, LockDecr, LockIncr, NodeNext, NodePrev, SetVal, SubNext, SubPrev, TP, Val
- Two variants: Easy API & Simple API
  - Developed in consultation with and intuitive to Go programmers
- Full set of utility functions



# Simulated Balance Transfers – M

```
baltrans(fromacct,toacct,amount)
  new frombalance          ; local variables used in thsi routine
  tstart ()                ; no local variables to be restored on TP restart
  set frombalance=^balance(fromacct) ; cache global in a local for performance
  set:amount>frombalance $code=",U123," ; check for sufficient funds; raise error if not
  set ^balance(fromacct)=frombalance-amount
  set ^balance(toacct)=^balance(toacct)+amount
  tcommit
  quit

main new i,total
  set ^balance("checking")=100
  set ^balance("savings")=1000
  set count=100
  set total=^balance("checking")+^balance("savings")
  write "Before: Checking=",^balance("checking")," Savings=",^balance("savings")," Total=",total,!
  for i=1:1:count job baltrans("savings","checking",$random(10)):(error="/dev/null":output="/dev/null")
  write i," jobs launched",!
  set total=^balance("checking")+^balance("savings")
  write "After: Checking=",^balance("checking")," Savings=",^balance("savings")," Total=",total,!
  quit
```

# Simulated Balance Transfers – Go

```
func baltrans(fromacct string, toacct string, amount int) {
    var errstr yottadb.BufferT
    errstr.Alloc(1024)
    yottadb.TpE(yottadb.NOTTP, &errstr, func(tpoken uint64, errstr *yottadb.BufferT) int32 {
        from_balance_s, _ := yottadb.ValE(tpoken, errstr, "^balance", []string{fromacct})
        from_balance, _ := strconv.Atoi(from_balance_s)
        if amount > from_balance {
            return yottadb.YDB_TP_ROLLBACK
        }
        from_balance -= amount
        yottadb.SetValE(tpoken, errstr, fmt.Sprintf("%d", from_balance), "^balance", []string{fromacct})
        to_balance_s, _ := yottadb.ValE(tpoken, errstr, "^balance", []string{toacct})
        to_balance, _ := strconv.Atoi(to_balance_s)
        to_balance += amount
        yottadb.SetValE(tpoken, errstr, fmt.Sprintf("%d", to_balance), "^balance", []string{toacct})
        return yottadb.YDB_OK
    }, "BATCH", nil);
}

func main() {
    defer yottadb.Exit()
    // Ensure ^balance("checking") and ^balance("savings") is set
    for i := 0; i < 100; i++ {
        go baltrans("checking", "savings", rand.Intn(100) - 50)
    }
}
```

# Moving Forward

- Not necessarily one correct way forward
  - Doubtful that MUMPS can evolve its way to world domination
  - Go is promising – performance, garbage collection, popularity, free / open source, use by Google, etc.
- To not evolve is to stagnate, and fade into irrelevance



YottaDB

*Thank You!*

K.S. Bhaskar  
bhaskar@yottadb.com

yottadb.com