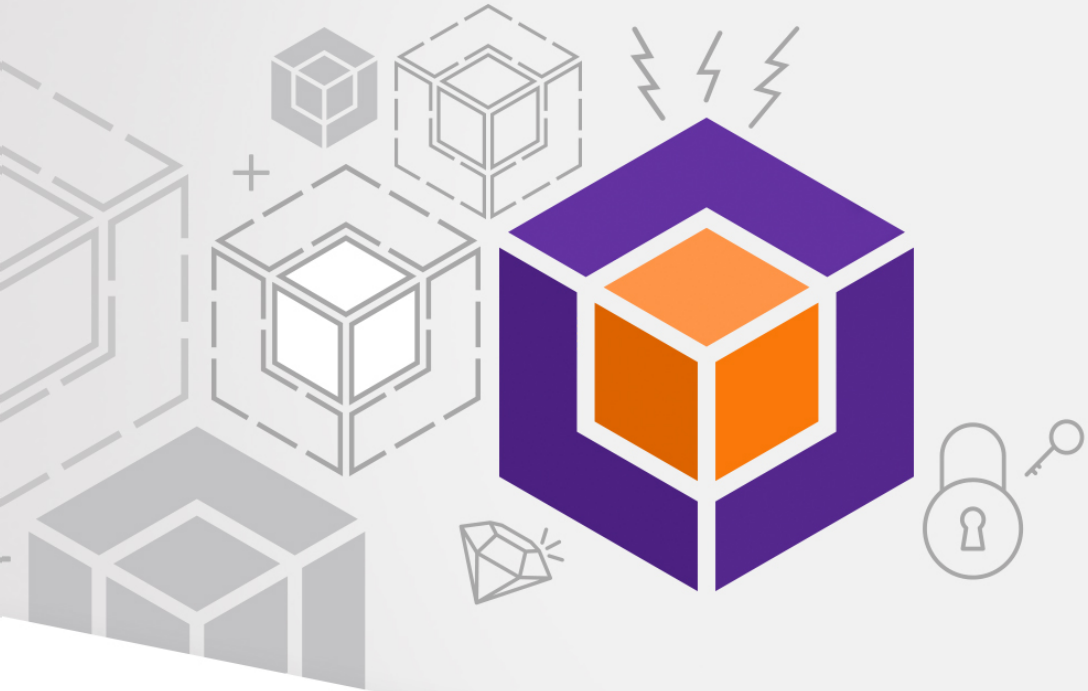
YottaDB

June 10, 2018



What Was Old Is New Again

YottaDB – https://yottadb.com

- A NoSQL database with a proven, mature code base that both scales up to enterprise-scale applications and scales down to the needs of embedded systems.
- *Rock Solid. Lightning Fast. Secure. Pick any three.*

Agenda

YottaDB

- The Past
  - Where are we and how did we get here?
- Making What was Old New Again
- The Future
- Demo
  - Still a work in progress

YottaDB

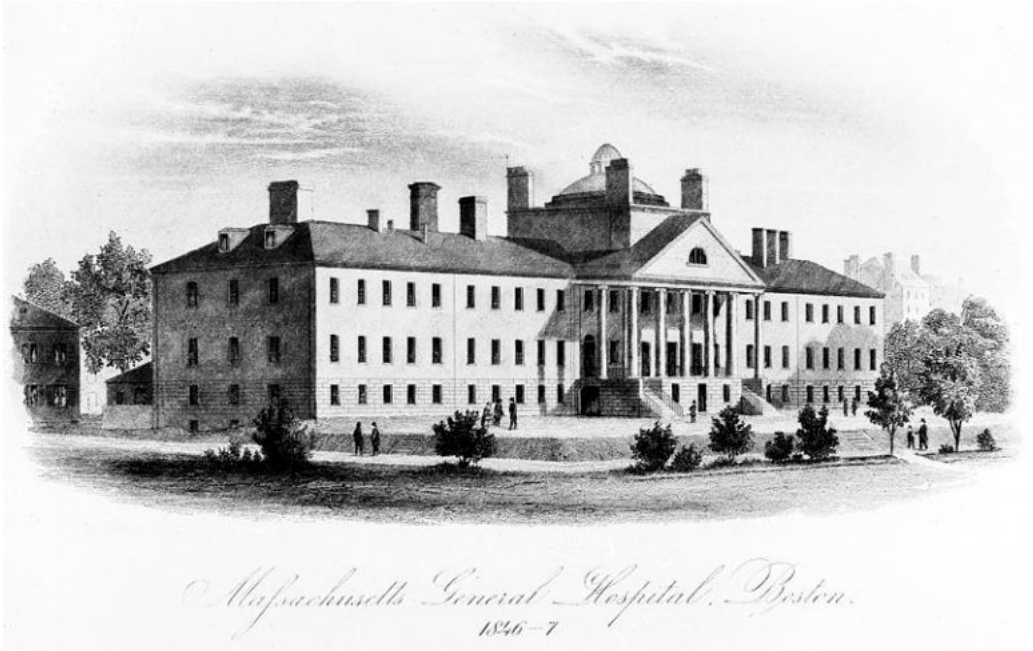
The Past

Where are we and how did we get here?

The Original Computer Database

YottaDB

- IBM Information Management System (IMS)
- Created to manage bill of materials & inventory of Saturn V & Apollo
  - Hierarchical data model – a NoSQL database!
- First released 1966; latest release 2017
- Runs on mainframe ⇒ Expen$ive


Massachusetts General Hospital, Boston.
1846–7

Massachusetts General Hospital, Boston

YottaDB

- Animal research laboratory circa 1966
  - Limited funding for computing
- Minicomputers – spare DEC PDP-7
- Accessible talent – across the river, in Cambridge
  - Massachusetts Institute of Technology
  - Bolt, Beranek and Newman

[M]UMPS

YottaDB

- Massachusetts General Hospital Utility Multi-Programming System
  - Operating system + hierarchical database file system + user interface + programming language + …
  - First used 1966/67
  - Ecosystem culture – user driven development; users and developers work closely together ⇒ pragmatic software without deep Computer Science theory
    - Not respected by CS academia

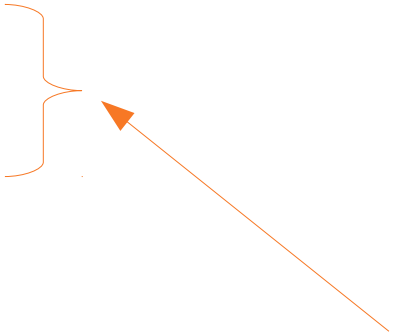
Key-Value Tuples

YottaDB

```
["Capital","Belgium","Brussels"]
["Capital","Thailand","Bangkok"]
["Capital","USA","Washington,DC"]
```

Key

Value



Always sorted – MUMPS means you never have to say you're sorting…

Schemaless

YottaDB

```
["Capital","Belgium","Brussels"]
["Capital","Thailand","Bangkok"]
["Capital","USA","Washington,DC"]
["Population","Belgium",13670000]
["Population","Thailand",84140000]
["Population","USA",325737000]
```

Schema determined entirely by application – MUMPS assigns no meaning

Numbers and strings (blobs) can be freely intermixed in values and keys except first

Default order for each key:

- Empty string ("")
- Canonical numbers in numeric order
- Strings (blobs) in lexical order

Mix Key Sizes

YottaDB

```
["Capital","Belgium","Brussels"]
["Capital","Thailand","Bangkok"]
["Capital","USA","Washington,DC"]
["Population","Belgium",13670000]
["Population","Thailand",84140000]
["Population","USA",325737000]
["Population","USA",17900802,3929326]
["Population","USA",18000804,5308483]
…
["Population","USA",20100401,308745538]
```

"Population" + 1 more key means value is latest population

"Population" + 2 more keys means value is population on date represented by last key

yyyymmdd

Keys ⟷ Array References

YottaDB

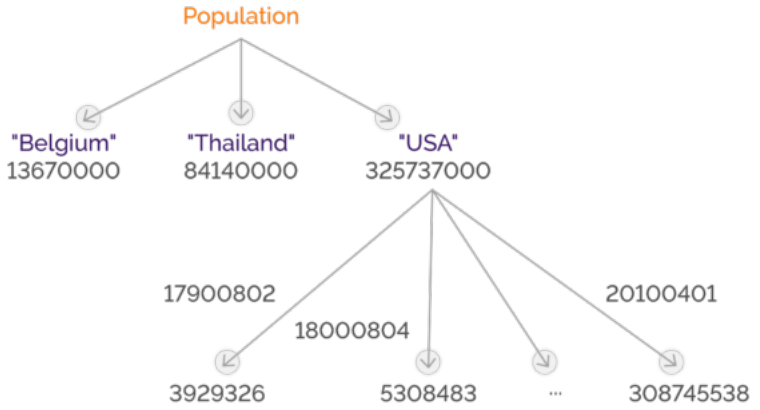
```
Population("Belgium")=13670000
Population("Thailand")=84140000
Population("USA")=325737000
Population("USA",17900802)=3929326
Population("USA",18000804)=5308483
…
Population("USA",20100401)=308745538
```

First key is variable name

Other keys are subscripts

Array references are a familiar programming paradigm

POPULATION TREE



Any JSON structure is representable as a tree, but not vice versa

Sharing and Persistence – Database Access YottaDB

- Process private, available only for lifetime of process

```
Population("Belgium")
Population("Thailand
Population("USA")
```

"local" variables

- Shared across processes, persistent beyond lifetime of any process

```
^Population("Belgium")
^Population("Thailand")
^Population("USA")
```

"global" variables

Spot the difference?

Universal NoSQL

YottaDB

- Satisfies common major NoSQL use cases
  - http://mgateway.com/docs/universalNoSQL.pdf
- NoSQL means “Not only SQL”

Noteworthy Features

YottaDB

- Tight binding of database to language
- Direct source code execution (initial implementation)
- Dynamic linking
- Multitasking
- Interactive / incremental usage
- Hierarchical locks (traffic light semantics)

Noteworthy Contempories

YottaDB

- C
- SQL
- TCP/IP
- UNIX

Evolution … 1

- 1970s
  - Language+database separate from operating system
- 1980s (GT.M – forerunner to YottaDB)
  - Programs are just text files in the file system
    - Compiled to object code for execution
    - While maintaining interactive / incremental usage

Evolution … 2

YottaDB

- 1990s
  - ACID transactions (GT.M)
  - Vendor consolidation
    - Just two commercial implementations left
- 2000s
  - GT.M/Linux moves to free / open source license

ACID Transactions

YottaDB

- Atomic – it all happens or none of it happens
- Consistent – logic inside a transaction cannot see internal state of another transaction
- Isolated – no other logic can see inside this transaction
- Durable – once committed, state change is permanent

ACID Transactions in GT.M/YottaDB

YottaDB

- Ensuring Consistency & Isolation with high concurrency is hard
- Optimistic Concurrency Control
  - http://daslab.seas.harvard.edu/reading-group/papers/kung.pdf
- Achieves high levels of concurrency & scalability
  - At the cost of a pathological case that application code must avoid

GT.M/YottaDB Today

- At the heart of mission-critical applications – the largest real-time core-banking and patient-centric healthcare systems in the world
- But not widely used in general purpose computing

Why Not Widely Used … 1

YottaDB

- Consequences of direct execution of source code
  - Needed to save memory and run fast
  - Single letter abbreviations of commands, short names

```
hello
  write "Hello, World!",!
  quit

hello w "Hello, World!",! q
```

Why Not Widely Used … 2

YottaDB

- Consequences of direct execution of source code
- Enterprise-scale applications on small computers
  - Expert friendly code, e.g.

```
S %P1=$S($L(%P1)>8:$E(%P1,1,8)-17000000_"."_$E(%P1,9,14),1:%P1-17000000)
;%P1 is now in FM format
I %P1["."",+$P(%P1,".",2)=0 S %P1=$$FMADD(+%P1,-1)_".24"
;If HL7 tz and local tz are the same
I %P2["L",%TZ=%LTZ S %P2=""
I (%P2["U")!(%P2["L"),%P1["." D  ;Build UCT from dat
. S %=$TR(%TZ,"+-","-+") ;Reverse the sign
. S %H=$E(%,1,3),%M=$E(%,1)_$E(%,4,5)
. S %P1=$$FMADD(%P1,,%H,%M) Q
```

Why Not Widely Used … 3

YottaDB

- Consequences of direct execution of source code
- Enterprise-scale applications on small computers
- Successful applications have long lives
  - Code written in the 1970s and 1980s was written to different standards of readability than code today
  - Application consistency for maintainability means coding style lags best practices for readability

Why Not Widely Used … 4

YottaDB

- Direct execution of source code
- Enterprise-scale applications on small computers
- Successful applications have long lives
- Vendor consolidation ended language evolution & standardization
  - One vendor able to acquire all implementations except GT.M

Why Not Widely Used … 5

YottaDB

- Direct execution of source code
- Enterprise-scale applications on small computers
- Successful applications have long lives
- Vendor consolidation ended language evolution & standardization
- Cultural issues inside and outside community

YottaDB

Making What was Old New Again

The Diamond is the Database

YottaDB

- Mature, proven code
  - “Rock Solid. Lighning Fast. Secure. Pick any three.”

The Language is What it is

YottaDB

- You either love it or you hate it
  - Like anchovies on your pizza
  - or like emacs vs. vi[m] vs. …
  - or like your religion vs. the other guy's religion
  - or…

YottaDB Strategy

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

Public domain from Wikimedia Commons

By GT1976 [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], from Wikimedia Commons

Photos are almost 100 years apart

YottaDB

From GT.M to YottaDB

Building on Strengths and
Accommodating What's New

Tight Database Binding is a Strength

YottaDB

- Create tight binding from database to C, just like the tight binding from database to the MUMPS language
- Make it as easy to use as any other library

```
source /usr/local/lib/yottadb/ydb_env_set
#include "libyottadb.h"
gcc -I $ydb_dist -L $ydb_dist -o myprog myprog.c -lyottadb
./myprog
```

Simple API – Key Functions

YottaDB

`ydb_data_s()` – determine whether node and/or subtree exist
`ydb_delete_s()` – delete node or both node & subtree
`ydb_delete_excl_s()` – delete all local variables (optionally except specified)
`ydb_get_s()` – get a value from a local or global variable node
`ydb_node_next_s()` – get next node (depth-first order)
`ydb_node_previous_s()` – get previous node
`ydb_set_s()` – set the value at a node
`ydb_subscript_next_s()` – get next subscript at deepest level (breadth-first order)
`ydb_subscript_previous_s()` – get previous subscript at deepest level
`ydb_tp_s()` – execute provided function with ACID transaction properties

Extend to Other Languages

YottaDB

- C is the *lingua franca* of computer languages
  - JavaScript wrapper – next major version to use Simple API
    - https://github.com/dlwicksell/nodem
  - Go is coming
  - SQL for reporting and to leverage other tools
    - Expert-friendly FOSS SQL/JDBC engine exists; make it user friendly
  - C++, Python, Rust, Java

Driven by user input and funding

Extend Platforms

YottaDB

- Linux on 32-bit ARM
  - ARMv7-A (e.g., Raspberry Pi 3, BeagleBone Black) added 2017
  - ARMv6 added 2018 (e.g., Raspberry Pi Zero)
- Linux on 64-bit ARM
  - Anticipated late 2018

YottaDB

The Future

“YottaDB Everywhere”

- Footprint fits in embedded systems
- Scales up to manage very large databases
- And everything in-between
- *“Rock solid. Lightning fast. Secure. Pick any three.”*

Everywhere

YottaDB Initial Targets

YottaDB

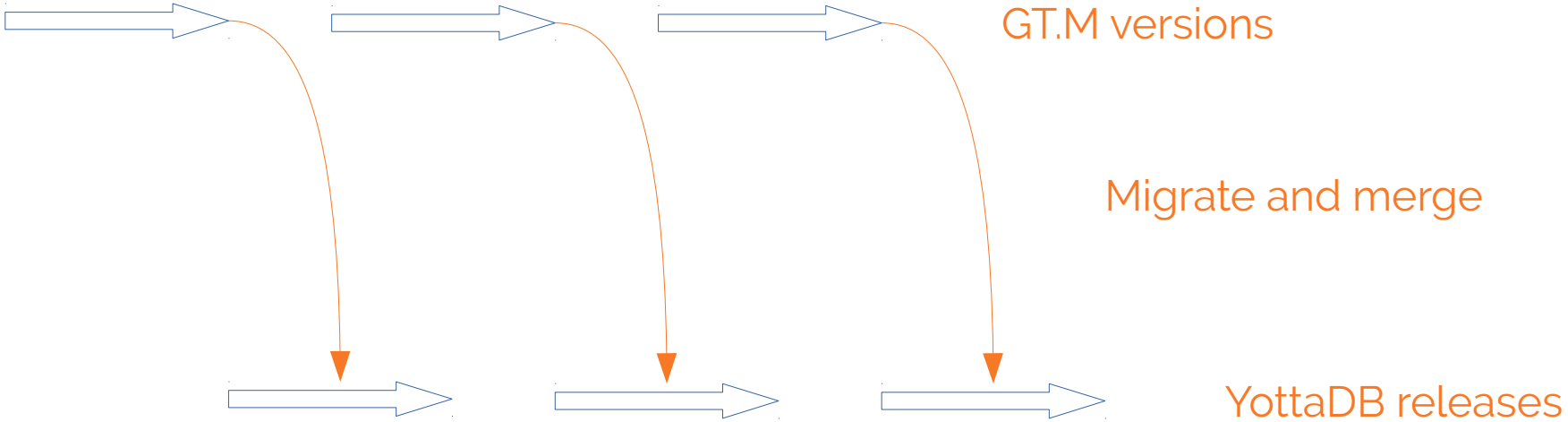
- Traditional GT.M applications
  - Including core banking and electronic health records
- “Big data”
  - e.g., alternative to Hadoop
- Internet of Things (IoT)
  - One database for the complete stack

Traditional – A Better GT.M … 1

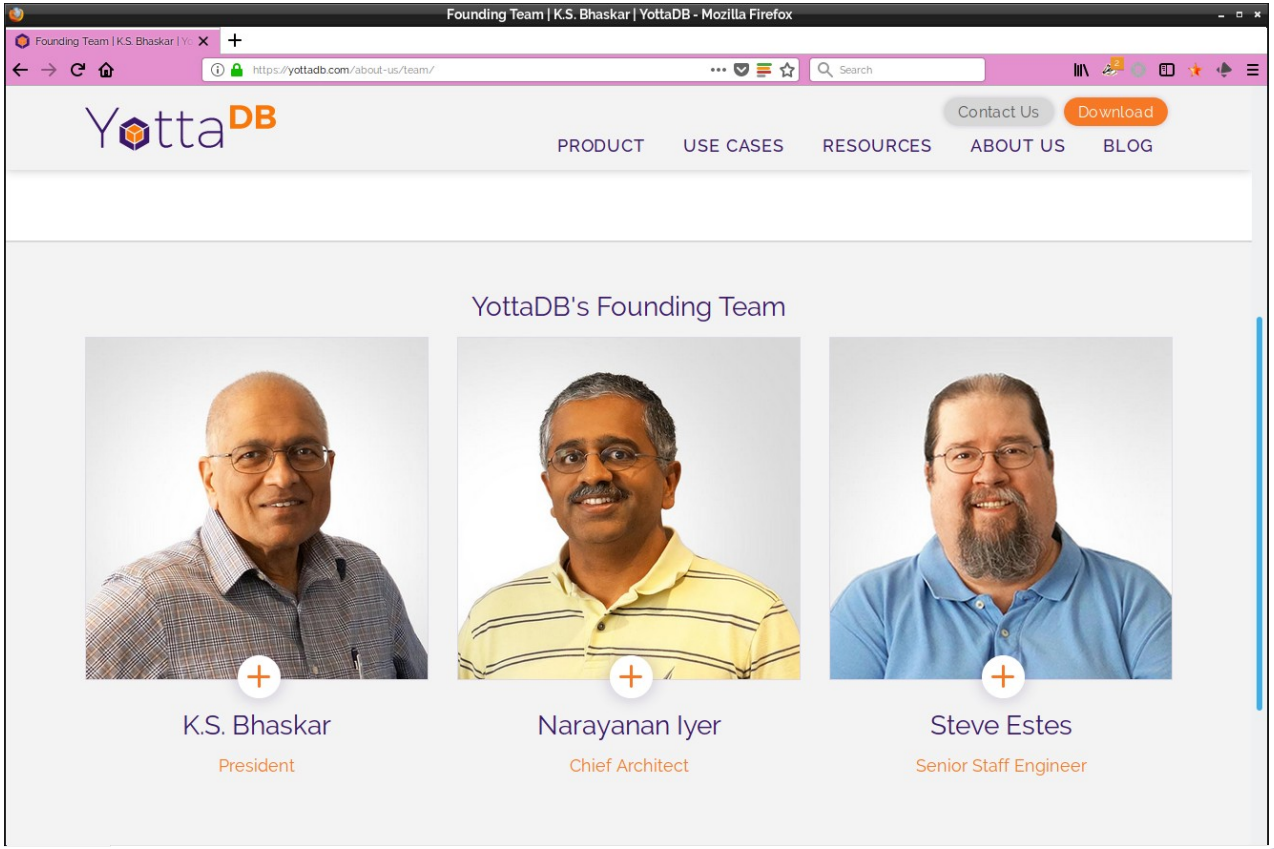
YottaDB

- Stay upward compatible
  - GT.M enhancements & fixes merged into code base

Close Relationship to GT.M is a Strength



Ensuring Upward Compatibility



More than 20 years experience working together on GT.M

Traditional – A Better GT.M … 2

YottaDB

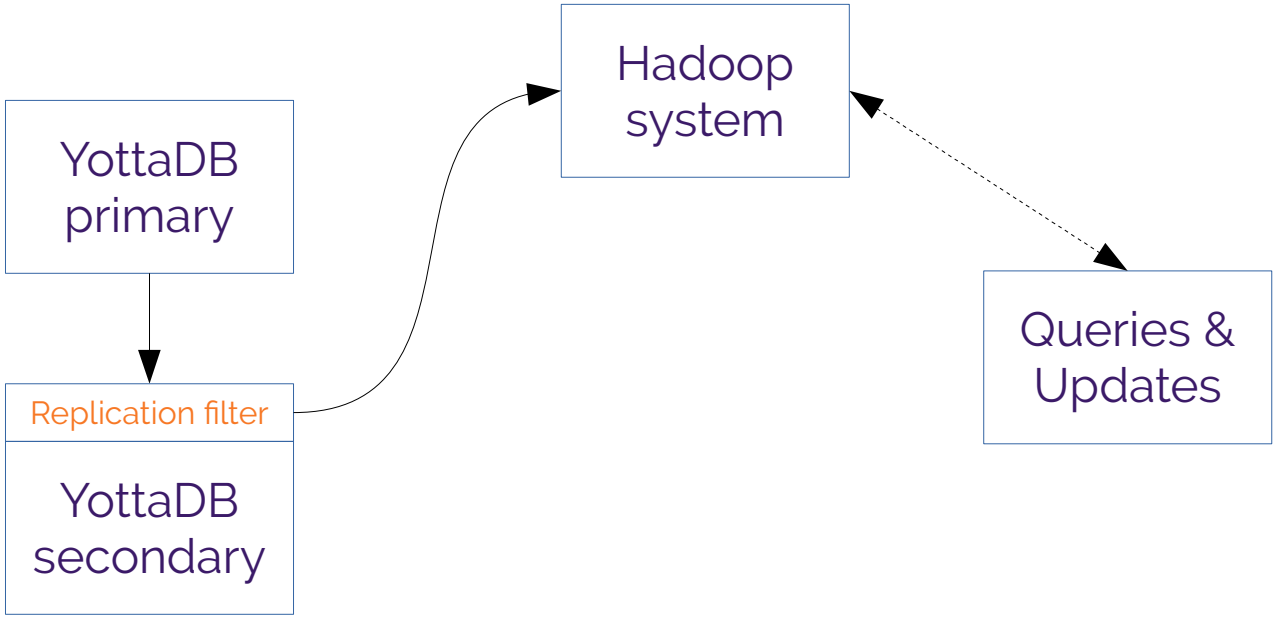
- GT.M enhancements & fixes merged into code base
- Performance & scalability specific to YottaDB
- Fixes specific to YottaDB

Big Data Approach

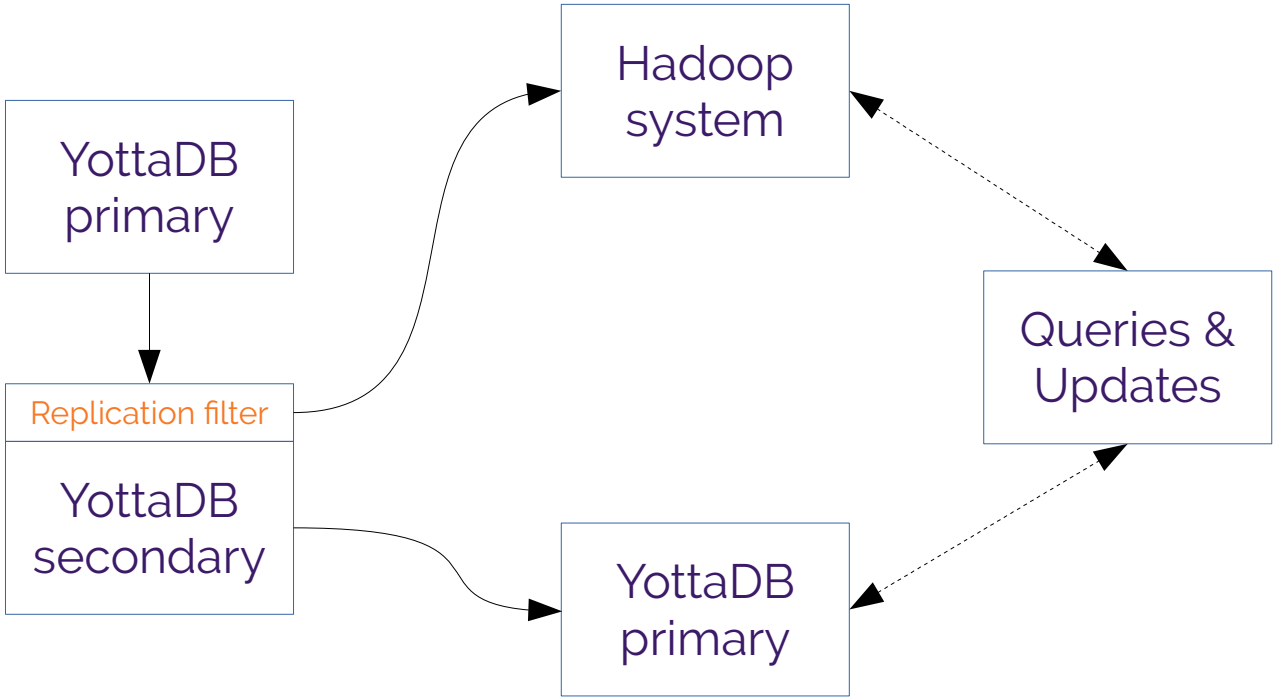
YottaDB

- Create Hadoop system from real-time banking application
  - Real time data feed replication stream → Hadoop
  - Write queries for working application
- Replace Hadoop database & application with YottaDB
- Incrementally expand functionality to more Hadoop applications and perhaps others (e.g., R)

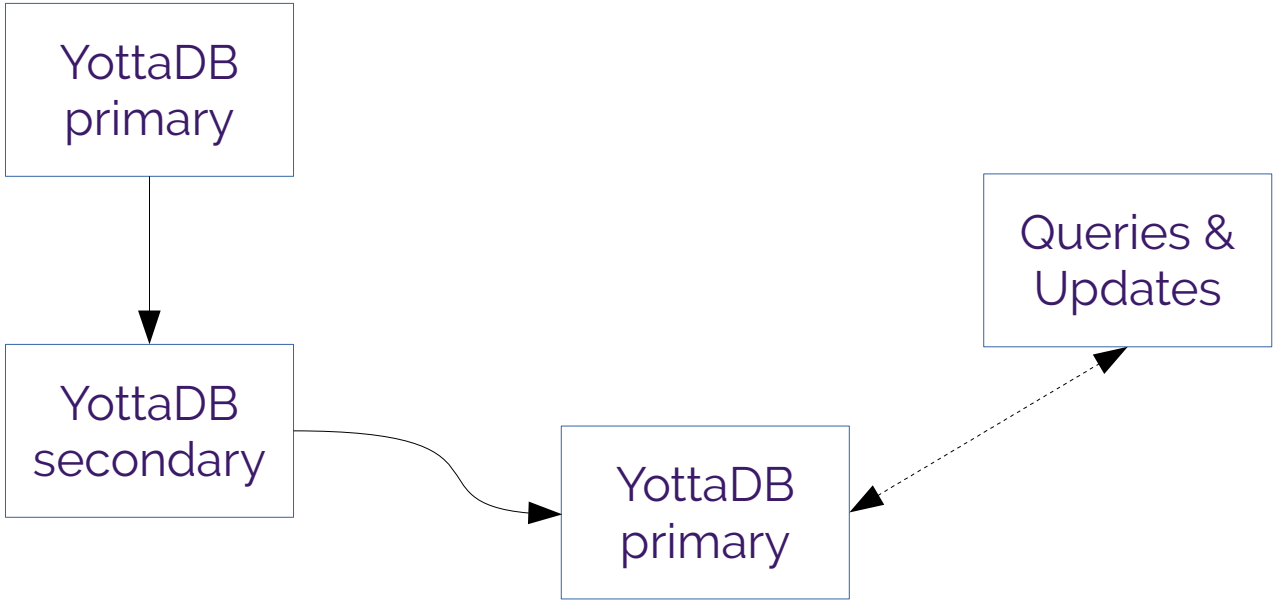
A Picture is Worth 1E3 Words … 1



A Picture is Worth 1E3 Words .. 2



A Picture is Worth 1E3 Words … 3



YottaDB

Demo

Still a work in progress

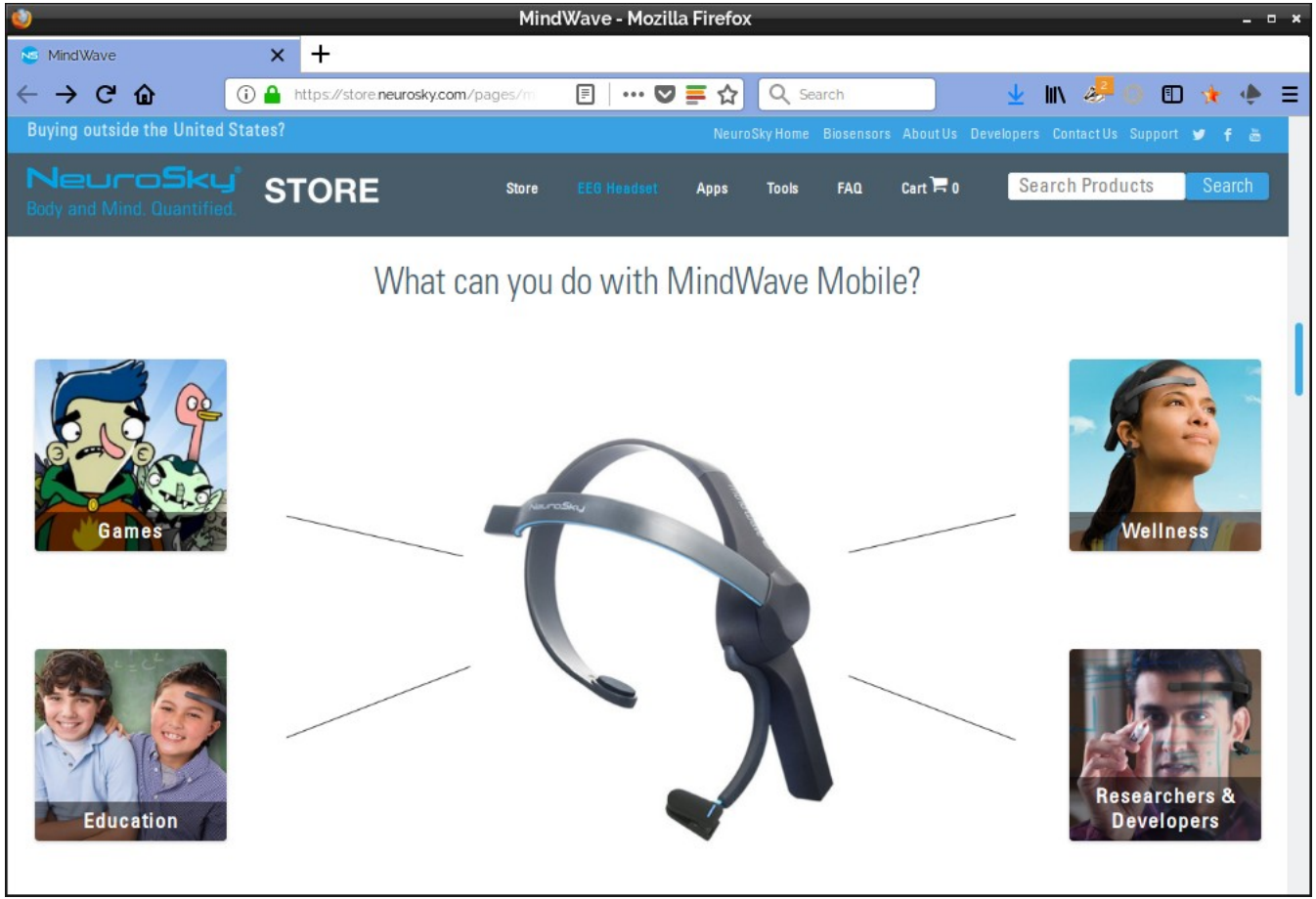
Goal

YottaDB

- Demonstrate YottaDB as a single database used on the edge and in the cloud

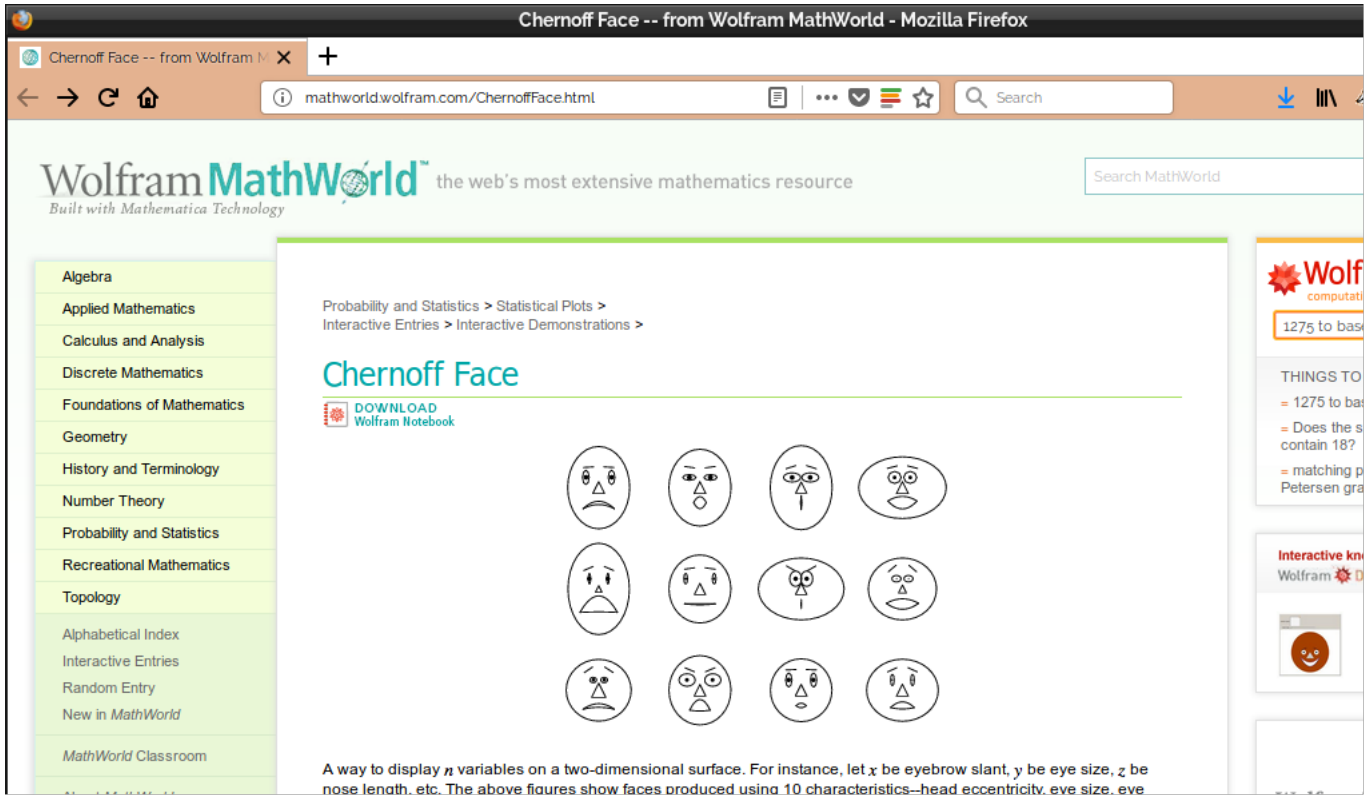
Internet of Things Demo – EEG Sensor

YottaDB



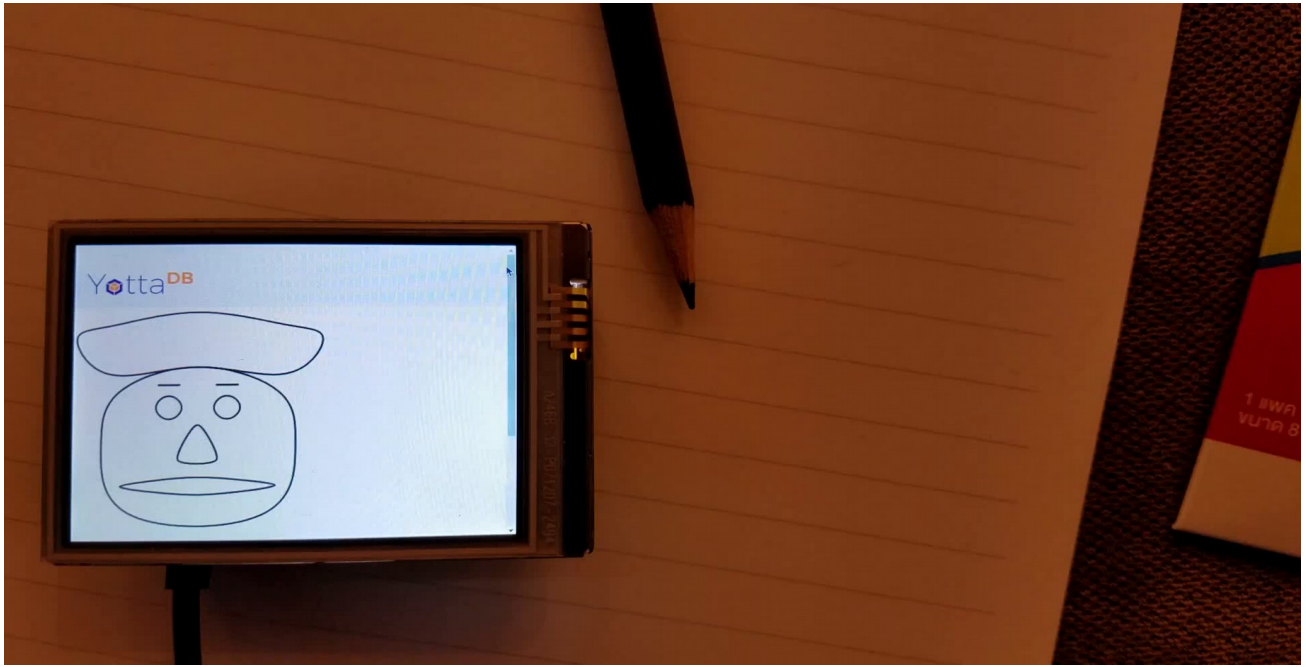
Internet of Things Demo – Chernoff Faces

YottaDB



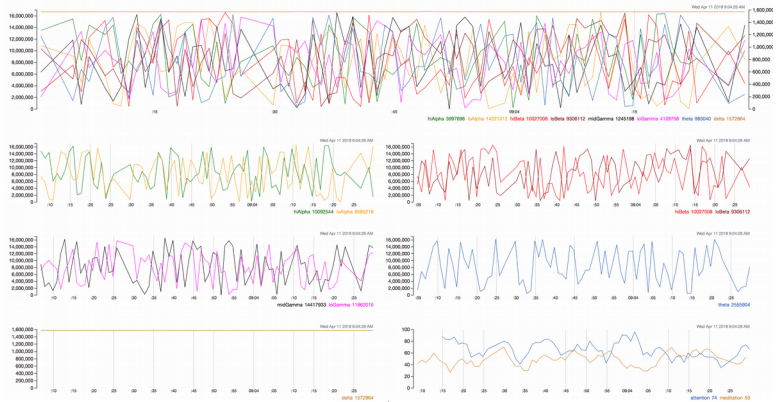
http://mathworld.wolfram.com/ChernoffFace.html

Chernoff Face Reading My Mind



Internet of Things Demo – Block Diagram

YottaDB



EEG Sensor

Bluetooth

YottaDB everywhere except EEG sensor

Raspberry Pi Zero W

Edge

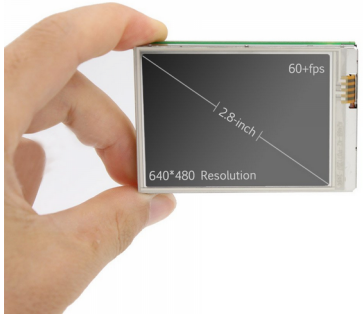
Replication

Graphing / Monitoring

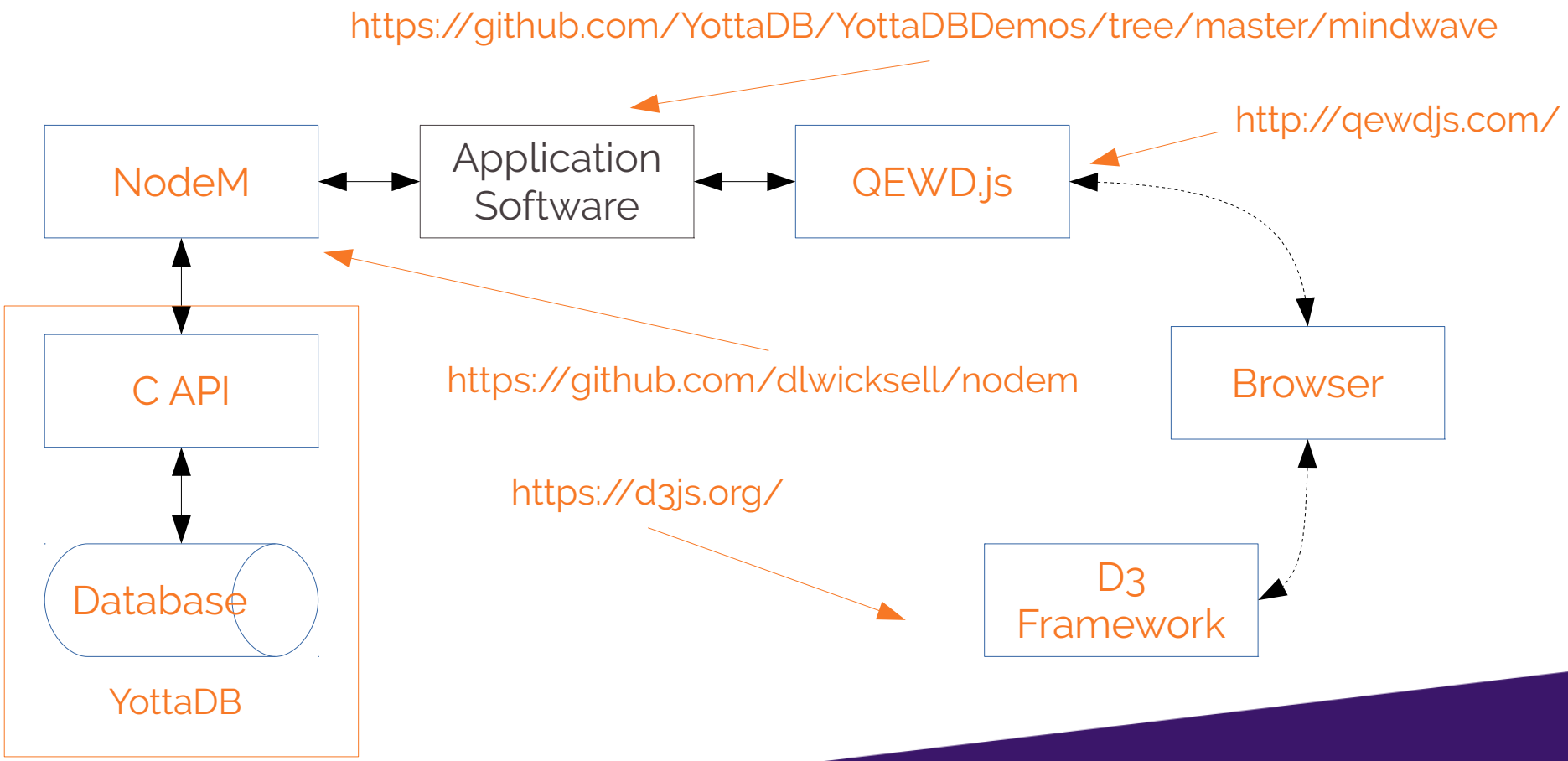
Triggers

EHR

Cloud



Demo Technology – All FOSS



Links

YottaDB

- Web site – https://yottadb.com
- Quick start – https://docs.yottadb.com/MultiLangProgGuide/MultiLangProgGuide.html#quick-start
- User documentation – https://yottadb.com/resources/documentation/
- Blog - https://yottadb.com/blog/
- Contact – K.S. Bhaskar / bhaskar@yottadb.com

YottaDB

Thank You!

yottadb.com