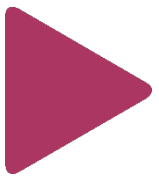
YottaDB™ Foundation

Comsan Channma (Neo)

T.N. Incorporation Ltd.



TN

About Me

- Department Manager, SE-Excellence Performance Lab Team, T.N. Incorporation Ltd.
- 17 years' experience on GT.M and YottaDB database (PROFILE CBS Project).
- SE Data Engineer Team Leader.
- Performance Testing Specialist.

Agenda

1st Day.

- Database Principle
- Introduction to YottaDB
- YottaDB User Command
- YottaDB Administrator Command
- YottaDB Installation
- YottaDB DB Structure

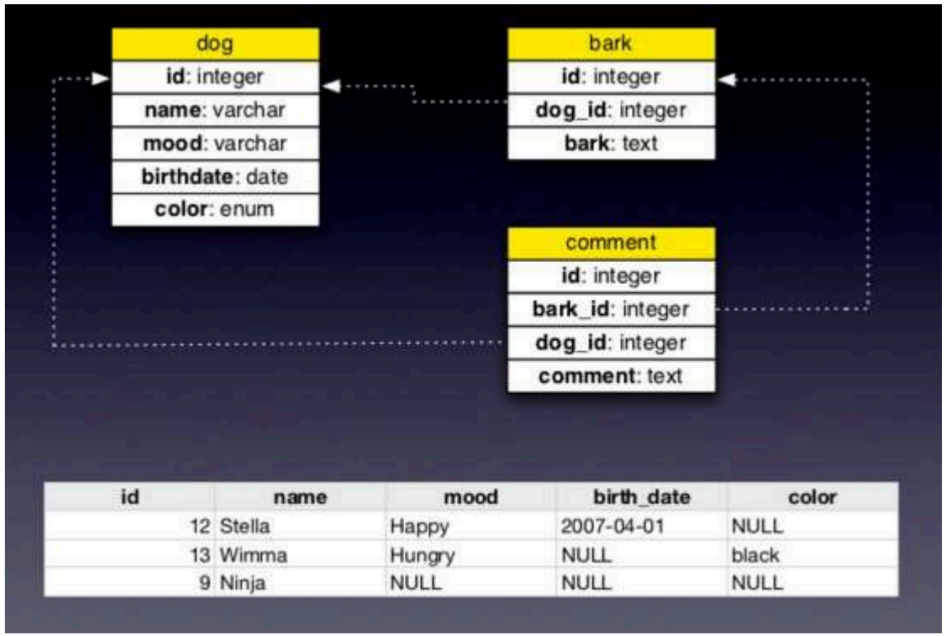
Database Principle (SQL VS. NoSQL)

Relational and non-relational defined

- Relational databases (RDBMS, SQL Databases)
  - Example: Microsoft SQL Server, Oracle Database, IBM DB2, MySQL
  - Mostly used in large enterprise scenarios
  - Analytical RDBMS (OLAP, MPP) solutions are Analytics Platform System, Teradata, Netezza
- Non-relational databases (NoSQL databases)
  - Example: YottaDB, Azure Cosmos DB, MongoDB, Cassandra
  - Four categories: Key-value stores, Wide-column stores, Document stores and Graph stores

Relational Store

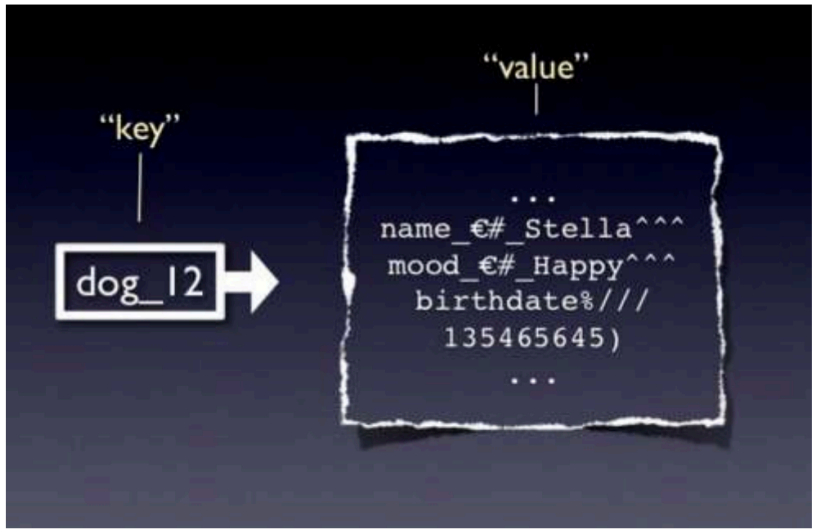
- Data stored in tables.
- Tables contain some number of columns, each of a type.
- A schema describes the columns each table can have.
- Every table's data is stored in one or more rows.
- Each row contains a value for every column in that table.
- Rows aren't kept in any particular order.



id	name	mood	birth_date	color
12	Stella	Happy	2007-04-01	NULL
13	Wimma	Hungry	NULL	black
9	Ninja	NULL	NULL	NULL

Key-Value Stores

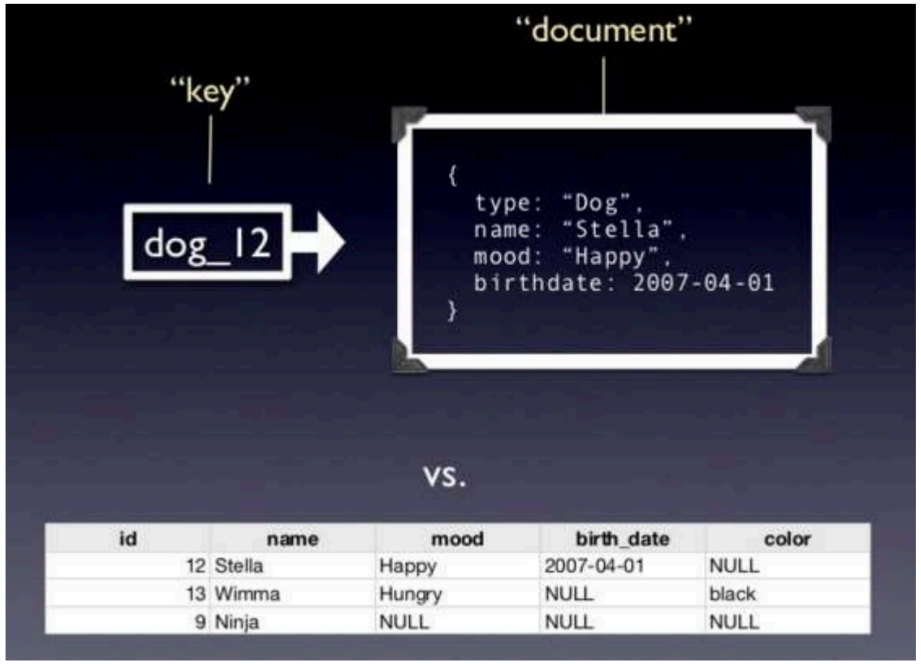
- Key -value stores offer very high speed via the least complicated data model
- anything can be stored as a value, as long as each value is associated with a key or name.

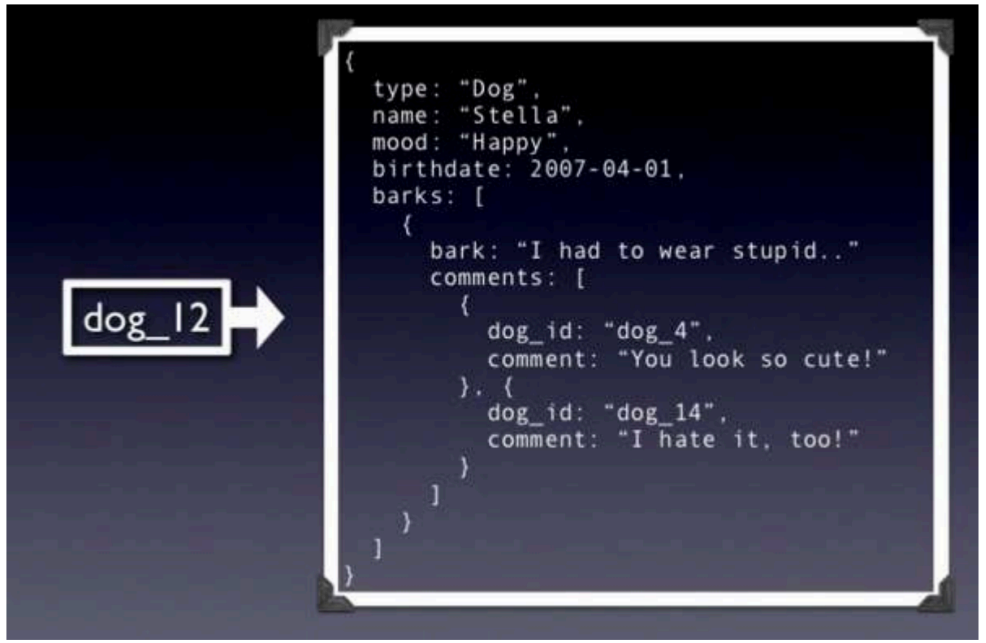


Key "dog_12": value_name "Stella", value_mood "Happy", etc

Document Stores

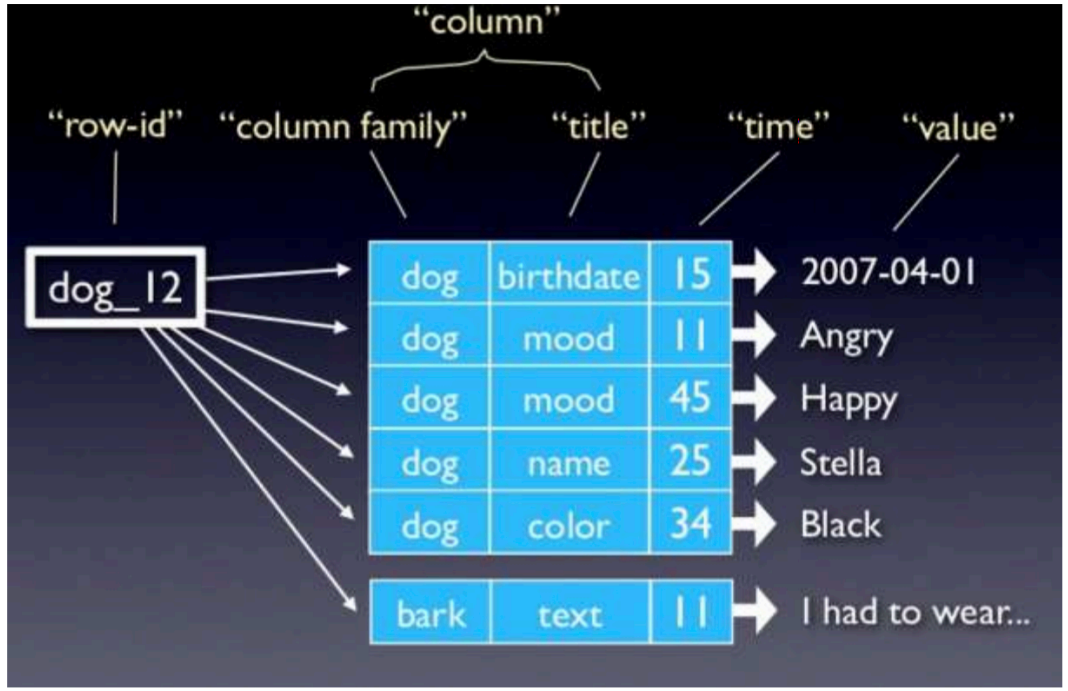
- Document stores contain data objects that are inherently hierarchical, tree-like structures (most notably JSON or XML). Not Word documents!





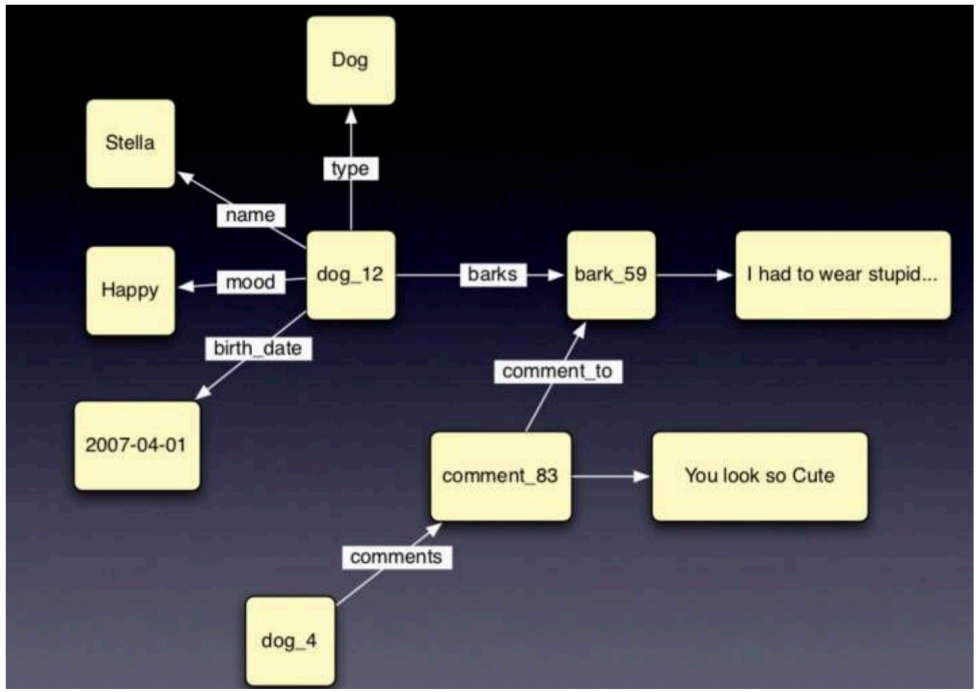
Wide-Column Stores

- Wide-column stores are fast and can be nearly as simple as key-value stores.
- They include a primary key, an optional secondary key, and anything stored as a value. Also called column stores



Graph Stores

- Schema-less, based on graph theory.
- Only two types of data inside the graph database.
  - Node and relationship (edge) .
- Widely use in social network system and large scale website architecture.



NoSQL Database

Key Value Store

Based on Amazon's **Dynamo** platform**:**
Highly Available Key-Value Store
Data Model:
- Global key-value mapping
- Big scalable HashMap
- Highly fault tolerant

Examples:
- Redis, Riak, YottDB

Column Family (KKV)

Based on **BigTable**: Google's Distributed Storage System for Structured Data
Data Model:
- A big table, with column families
- Map Reduce for querying/processing
- Every row can have its own Schema

Examples:
- HBase, Cassandra, MapR DB

Document Database

Based on Lotus Notes
Data Model:
- A collection of documents
- A document is a key value collection
- Index-centric, lots of map-reduce

Examples:
- CouchDB, MongoDB

Graph Database

Based on Euler & Graph Theory
Data Model:
- Nodes and Relationships

Examples:
- Neo4j, OrientDB, InfiniteGraph, DSE Cassandra

Relational Database : Pros

- Works with structured data
- Supports strict ACID transactional consistency
- Supports joins
- Built-in data integrity
- Large eco-system
- Relationships via constraints
- Limitless indexing
- Strong SQL
- OLTP and OLAP
- Most off-the-shelf applications run on RDBMS

Relational Database : Cons

- Does not scale out horizontally (concurrency and data size) – only vertically, unless use sharding
- Data is normalized, meaning lots of joins, affecting speed
- Difficulty in working with semi-structured data
- Schema-on-write
- Cost

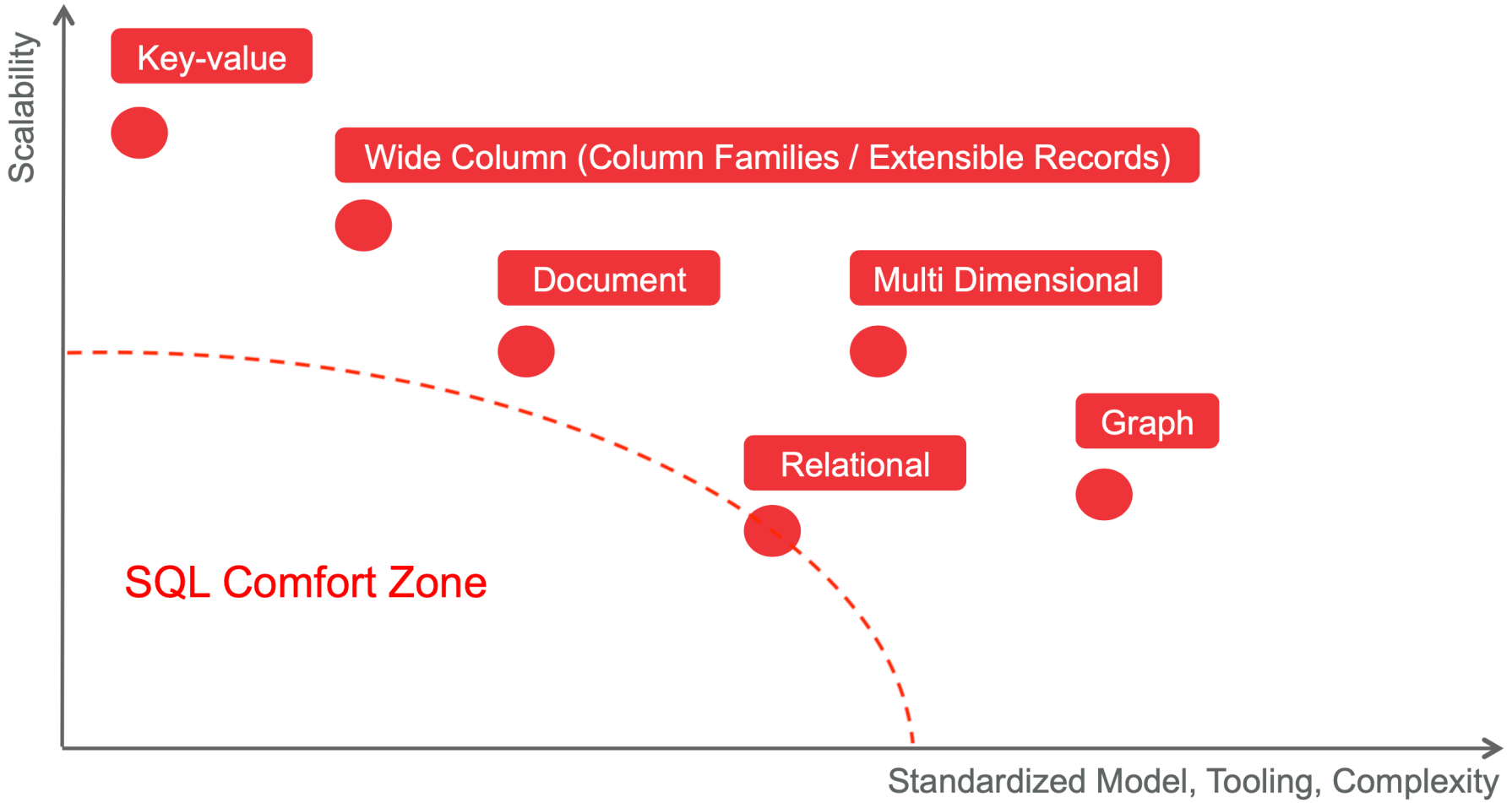
NoSQL : Pros

- Works with semi-structured data (JSON, XML)
- Scales out (horizontal scaling – parallel query performance, replication)
- High concurrency, high volume random reads and writes
- Massive data stores
- Schema-free, schema-on-read
- Supports documents with different fields
- High availability
- Cost
- Simplicity of design: no “impedance mismatch”
- Finer control over availability
- Speed, due to not having to join tables

NoSQL : Cons

- Weaker or eventual consistency (BASE) instead of ACID
- Limited support for joins, does not support star schema
- Data is denormalized, requiring mass updates (i.e. product name change)
- Does not have built-in data integrity (must do in code)
- No relationship enforcement
- Limited indexing
- Weak SQL
- Limited transaction support
- Slow mass updates
- Uses 10-50x more space (replication, denormalized, documents)
- Difficulty tracking schema changes over time
- Most NoSQL databases are still too immature for reliable enterprise operational applications

Data store Positioning



Introduction to YottaDB

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.

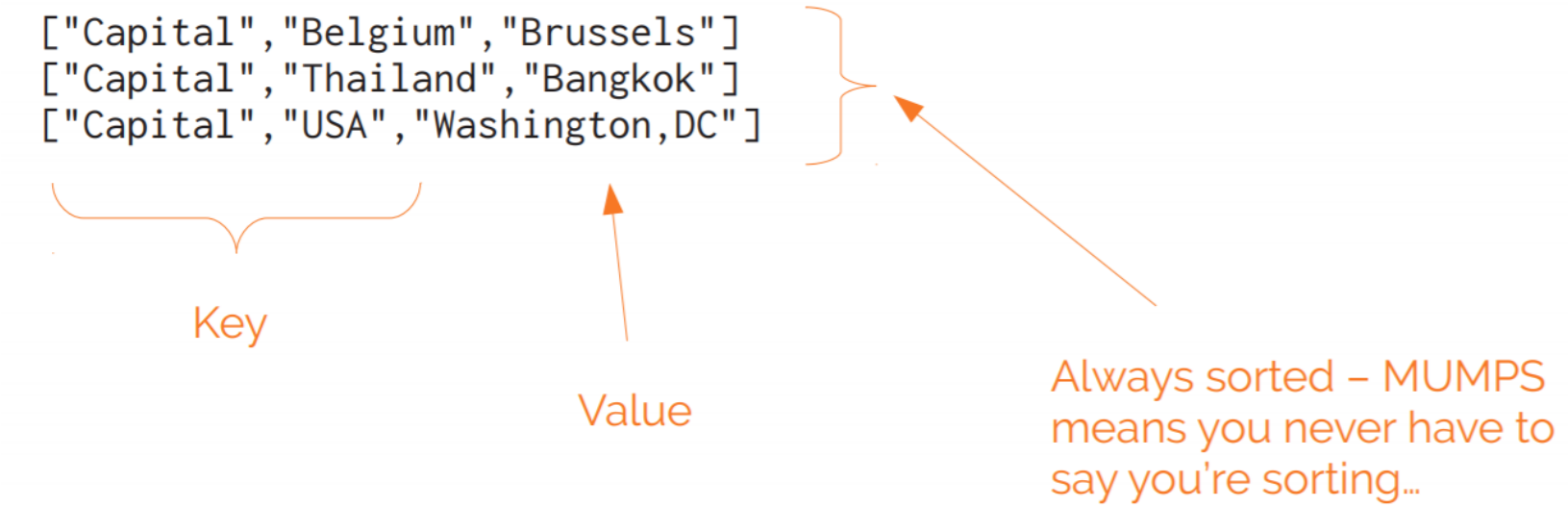
YottaDB is a NoSQL database

- Data is stored as a hierarchy of key-value data
- i.e., [“people”, “sanchez”, “rick”, “alive”]=”?”
- Very good for data which has hierarchy
- Where in SQL you would have a related table, you instead represent it as a “subscript” of the parent key

YottaDB Characteristics

- Data is “type-less”
- Relationships can be described as:
- Hierarchical
- Multi-dimensional sparse arrays
- Content-associative memory
- Daemon-less
- Built-in M language (compiled and interpret)
- Transaction processing support (ACID)
- Very fast database operation (read / write)

Key-Value Tuples



Schemaless

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

Default order for each key:

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

Schema determined entirely by application – MUMPS assigns no meaning

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

Mix Key Sizes

```
["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]
```

yyyymmdd

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

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

Keys Array References

```
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

- 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?

Multi-dimensional sparse arrays

Physical YottaDB database record example

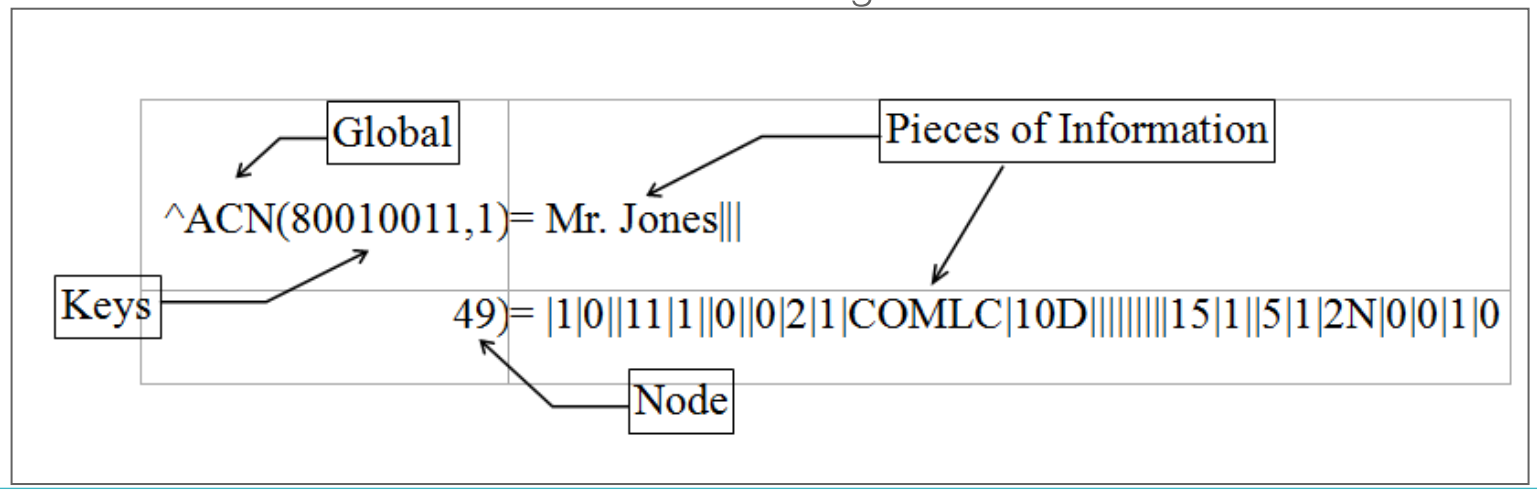
```
^ACN(19,1)="สมชาย สายชม"
^ACN(19,49)="|0|3||13||||0|||3|||0|||||||||0||||||||||||1|2|0|||||1|1|1"
^ACN(19,50)="2001|D|SAV||1|สายชม,สมชาย|200000|0|0||0|THB|1||SAV1|0|0|0|||20
AA|KTB|||0|0|0||0|||0"
^ACN(19,51)="0|60383|||5|6|0|0|||||0|59931|1|||0||60383|0|||||||||0||0||1"
^ACN(19,52)="0|59791||||0.75000|||||||||||||||||0"
^ACN(19,53)="||100|||||||||||||||0"
^ACN(19,54)="0|0|60388||||-363.24|0||0|0|||0.00000|1|||0|||||0|||||||0|0.00000|
0|0.00000|0||||||0||0|||PER1||||||||||||0.00000|0||0.00000|0||||||||||0||0"
^ACN(19,55)="|0|0||||||0"
^ACN(19,57)=".75|2|60389|60388|60446|60265||||||||||||0|||||||||||||||||||60416|
60385||||||||||0"
^ACN(19,60)="SAV002|JUN30DEC31|||1DA|||||||||||||||||||||||||||||||||||||||||
||||||||1MAE|0|||99Y"
^ACN(19,61)="|||||||||||||||||||||||||||0||0||0"
```

Global

A global is a structured representation of the physical data located in the database. Profile uses the string method of global placement (i.e., multiple nodes hold multiple pieces of information). Pieces of information are located by global name and key and are separated by delimiters ("|").

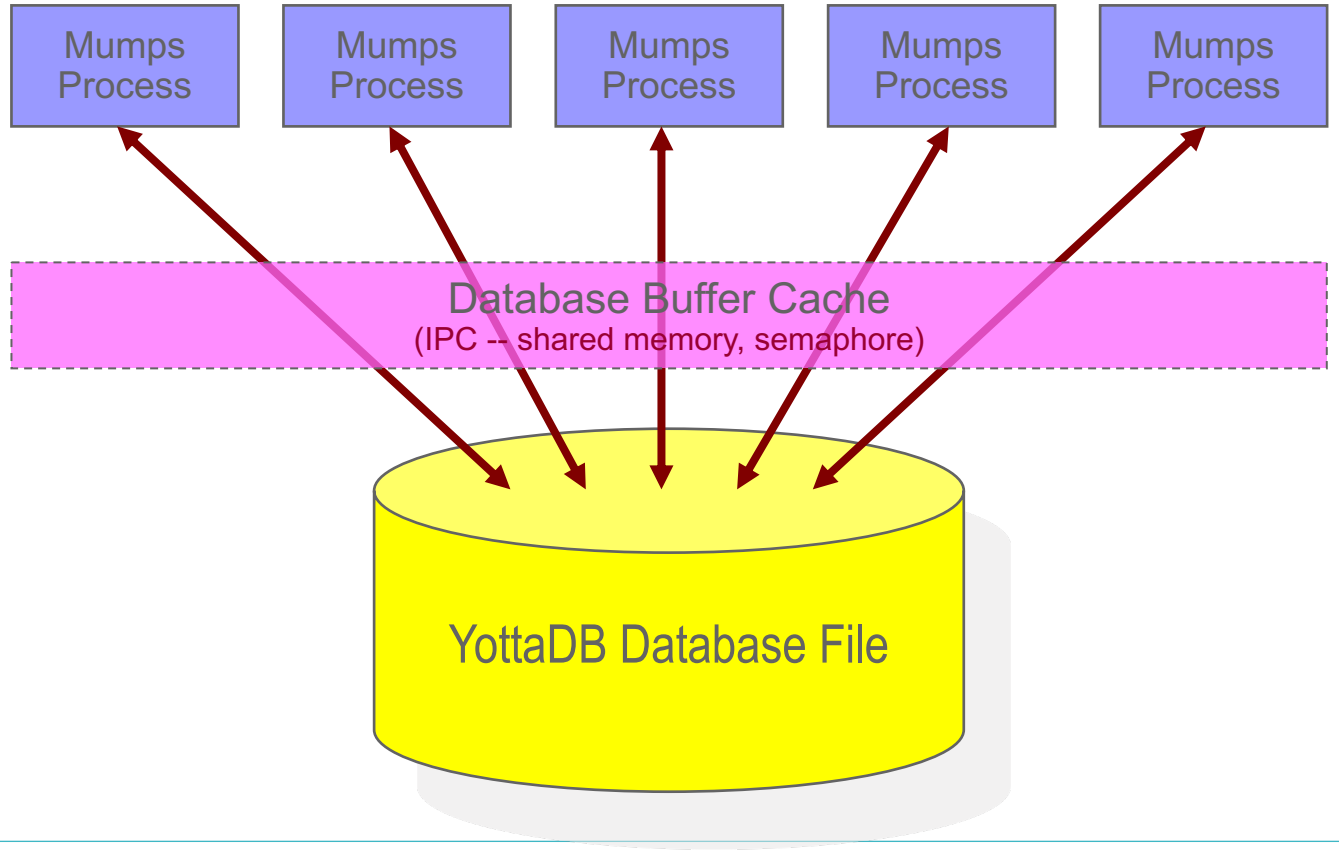
Example:

The first two nodes of an Account Record might look like this:



YottaDB is Daemon-less

Process that access YottaDB has to perform database processing logic



YottaDB User Command

YottaDB Access

- Use “yottadb” command or “mumps” command (GT.M traditional style) for YottaDB local access.
- Use M command (M programming language) to manage the data.

YottaDB Invocation Mode

- **-direct**: Invokes YottaDB in direct mode where you can enter M commands interactively.
- **<list of M source files>**: Invokes YottaDB in compiler mode, invoke YottaDB by entering a list of file names to compile as an argument. YottaDB then compiles the specified programs into .o files, UNIX shell globbing to resolve wild-cards (* and ?) in names.
- **-run ^routine_name**: -r invokes YottaDB in auto-start mode. The second argument is taken to be an M entryref, and that routine is automatically executed, bypassing direct mode. Depending on your shell, you may need to put the entryref in quotes.

YottaDB User Command Example

- Access and exit from YDB Prompt
- Check YottaDB Version
- List Global
- List mumps file and free space
- Show Global contents
- Set, Write and Kill Global
- Manipulate data in piece
- Merge Global

Access to YDB Prompt

Access YDB Prompt

```
$ cd /ydbdir
$ ../ydbenv
$ yottadb -direct

YDB>
```

Exit from YDB Prompt

```
YDB>HALT
$
```

```
YDB>H
$
```

Check YottaDB Version

```
YDB>WRITE $ZVERSION

YDB>W $ZVER
```

List Global

```
YDB> d ^%G

Output device: <terminal>:

List ^?D

Global Directory

Global ^*
^%ydboctoocto       ^%ydboctoschema
Total of 2 globals.

Global ^
```

List mumps files and free space

```
YDB> d ^%FREECNT
Region          Free     Total          Database file
------          ----     -----          -------------
DATA              94       100 ( 94.0%) /ydbdir/gbls/mumps.data
OCTO              56       100 ( 56.0%) /ydbdir/gbls/mumps.octo

YDB>
```

Set Global value

```
YDB>S ^ZTEST(1)="one"
YDB>S ^ZTEST(2)="two"
YDB>S ^ZTEST(1,1)="one,one"
YDB>S ^ZTEST(1,2)="one,2"
YDB>S ^ZTEST(300)=300
YDB>S ^ZTEST("FIVE")=555
YDB>S ^ZTEST="ABCDEFG"

YDB>ZWR ^ZTEST
^ZTEST="ABCDEFG"
^ZTEST(1)="one"
^ZTEST(1,1)="one,one"
^ZTEST(1,2)="one,2"
^ZTEST(2)="two"
^ZTEST(300)=300
^ZTEST("FIVE")=555
```

Show Global Contents

```
YDB> d ^%G

Output device: <terminal>:

List ^ZTEST
^ZTEST="ABCDEFG"
^ZTEST(1)="one"
^ZTEST(1,1)="one,one"
^ZTEST(1,2)="one,2"
^ZTEST(2)="two"
^ZTEST(300)=300
^ZTEST("FIVE")=555

List ^
```

Another ways to show Global

Write Command

```
YDB> W ^ZTEST(1,1)
one,one
```

ZWR Command

```
YDB> ZWR ^ZTEST(1,1)
^ZTEST(1,1)="one,one"
```

```
YDB> ZWR ^ZTEST(1,*)
^ZTEST(1)="one"
^ZTEST(1,1)="one,one"
^ZTEST(1,2)="one,2"
```

Replace Global value

```
YDB>S ^ZTEST(2)=22222

YDB>ZWR ^ZTEST
^ZTEST="ABCDEFG"
^ZTEST(1)="one"
^ZTEST(1,1)="one,one"
^ZTEST(1,2)="one,2"
^ZTEST(2)=22222
^ZTEST(300)=300
^ZTEST("FIVE")=555
```

Kill Global

```
YDB>K ^ZTEST(1,2)          ← Only node ^ZTEST(1,2) removed

YDB>ZWR ^ZTEST
^ZTEST="ABCDEFG"
^ZTEST(1)="one"
^ZTEST(1,1)="one,one"
^ZTEST(2)=22222
^ZTEST(300)=300
^ZTEST("FIVE")=555

YDB>K ^ZTEST(1)            ← All nodes under ^ZTEST(1) removed

YDB>ZWR ^ZTEST
^ZTEST="ABCDEFG"
^ZTEST(2)=22222
^ZTEST(300)=300
^ZTEST("FIVE")=555
```

Node and Piece

Set ^ZPIECE Global

```
YDB>S ^ZPIECE(1)="one|two|three|four|five"

YDB>W ^ZPIECE(1)
one|two|three|four|five
```

Show content in specific piece

```
YDB>W $P(^ZPIECE(1),"|",3)
three
YDB>W $P(^ZPIECE(1),"|",5)
five
```

Node and Piece

Change content in specific piece

```
YDB>W ^ZPIECE(1)
one|two|three|four|five

YDB>S $P(^ZPIECE(1),"|",4)="PIECE FOUR"

YDB>W ^ZPIECE(1)
one|two|three|PIECE FOUR|five
```

Merge Global

```
YDB>Set ^gbl1="one"
YDB>Set ^gbl1(1,1)="oneone"
YDB>Set ^gbl1(1,1,3)="oneonethree"
YDB>Set ^gbl1(1,2,4)="onetwofour"
YDB>Set ^gbl2(2)="gbl2_2"
YDB>Set ^gbl2(2,1,3)="gbl2_2_1_3"
YDB>Set ^gbl2(2,1,4,5)="gbl2_2_1_4_5"
YDB>Merge ^gbl1(1)=^gbl2(2)
YDB>WRITE $Reference
^gbl1(1)
YDB>ZWRite ^gbl1
^gbl1="one"
^gbl1(1)="gbl2_2"
^gbl1(1,1)="oneone"
^gbl1(1,1,3)="gbl2_2_1_3"
^gbl1(1,1,4,5)="gbl2_2_1_4_5"
^gbl1(1,2,4)="onetwofour"
YDB>ZWRITE ^gbl2
^gbl2(2)="gbl2_2"
^gbl2(2,1,3)="gbl2_2_1_3"
^gbl2(2,1,4,5)="gbl2_2_1_4_5"
YDB>
```

M Language Programmer's Guide

- https://docs.yottadb.com/ProgrammersGuide/man.html

YottaDB Administrator Command

YottaDB Utility Programs

1. **MUPIP** (M Peripheral Interchange Program) is the YottaDB utility program for general database operations, YottaDB Journaling, Multi-site Database Replication, and some non-database operations.

2. **DSE** (The Database Structure Editor) is the YottaDB utility program to examine and alter the internal database structures. DSE edits YottaDB Database Structure (GDS) files. It provides an extensive database “patch” facility (including block integrity checks), searches for block numbers and nodes, and provides symbolic examination and manipulation facilities.

3. **GDE** (The Global Directory Editor) is a YottaDB utility program that creates and maintains global directories. GDE provides commands for operating on the global directory.

4. **LKE** (The M Lock Utility) is the YottaDB utility program that examines and modifies the lock space where YottaDB maintains the current M LOCK state. LKE can monitor the locking mechanism and remove locks.

M Peripheral Interchange Program (MUPIP)

- The M Peripheral Interchange Program (MUPIP) is a utility that provides an assortment of tools for YDB database management.
- Once the YDB environment is defined, MUPIP may be used either in direct mode or at a MUPIP prompt
- To get to a MUPIP prompt execute the following instruction:
  - $ $ydb_dist/mupip
  - MUPIP>
  - Or
  - $ $ydb_dist/mupip stop 1158

Mupip Operations - Standalone and Concurrent Access

Operations	MUPIP Command	Database Access Requirements
Backup database files	MUPIP BACKUP	Backup never requires standalone access and concurrent write access is controlled by -[NO]ONLINE.
Create and initialize database files	MUPIP CREATE	Standalone Access
Convert a database file from one endian format to the other (BIG to LITTLE or LITTLE to BIG)	MUPIP ENDIANCVT	Standalone Access
Recover database files (for example, after a system crash) and extract journal records	MUPIP JOURNAL	Standalone Access
Restore databases from bytestream backup files	MUPIP RESTORE	Standalone access
Properly close database files when processes terminate abnormally.	MUPIP RUNDOWN	Standalone access
Modify database and/or journal file characteristics	MUPIP SET	Standalone access is required if the MUPIP SET command specifies -ACCESS_METHOD, -GLOBAL_BUFFERS, -MUTEX_SLOTS, -LOCK_SPACE or -NOJOURNAL, or if any of the -JOURNAL options ENABLE, DISABLE, or BUFFER_SIZE are specified.
Grow the size of BG database files	MUPIP EXTEND	Concurrent Access
Export data from database files into sequential (flat) or binary files	MUPIP EXTRACT	Although MUPIP EXTRACT command works with concurrent access, it implicitly freezes the database to prevent updates. Therefore, from an application standpoint, you might plan for a standalone access during a MUPIP EXTRACT operation.
Prevent updates to database files	MUPIP FREEZE	Standalone access.
Check the integrity of GDS databases	MUPIP INTEG	Concurrent access. However, standalone access is required if MUPIP INTEG specifies -FILE
Import data into databases	MUPIP LOAD	Although MUPIP LOAD works with concurrent access, you should always assess the significance of performing a MUPIP LOAD operation when an application is running because it may result in an inconsistent application state for the database.
Defragment database files to improve performance	MUPIP REORG	Concurrent access.
Send an asynchronous signal to a YottaDB process	MUPIP INTRPT	Non-database access.
Reports information related to relinkctl files and their associated shared memory segments.	MUPIP RCTLDUMP	Non-database access.
Stop YottaDB processes	MUPIP STOP	Non-database access.

MUPIP CREATE

- The CREATE command generates database files using the information stored in the Global Directory. The command uses the global directory to map a region to a segment and a segment to a file. CREATE also initializes the Greystone Technology Database Structure (GDS).
- Use the MUPIP CREATE command to create (1) a new database or (2) a new copy of a previously existing file during a database reorganization. If a database file already exists for the segment, CREATE takes no action.

MUPIP CREATE (Cont.)

- The format of the CREATE command is:

  MUPIP> CREATE -region

- Create takes only one qualifier. The optional -region qualifier specifies a single region for which to create a database file. By default, CREATE sets up database files for all regions in the current Global Directory that have no corresponding file
- To create a database file for the UBG region based upon the mumps.gld file:

  MUPIP> CREATE -REGION=DATA

  $ mupip create -region=DATA

  Created file /ydbinst/gbls/mumps.data

MUPIP EXTEND

- The EXTEND command increases the size of a database file. A database file will extend under normal use. There is usually no need to extend a file with the EXTEND command
- The format for the EXTEND command is:

  MUPIP > EXTEND –BLOCKS= [#] region name

  - After the blocks qualifier, enter the number of database blocks you want the database file to be extended. Also enter the region name that you want to extend

MUPIP EXTEND Example

- To extend the TBLS UBG database file by 1000 database blocks:

```
$ mupip extend -blocks=1000 DATA

        Extension successful, file /educ9/gbls/mumps.data extended by
        1000 blocks.  Total blocks = 22024
```

MUPIP RUNDOWN

- When database files have not be properly closed, RUNDOWN closes currently inactive databases and releases the central memory they claim
  - In normal operation, the last process to close a database file performs the RUNDOWN actions.
- The format of the INTEG command is:

  MUPIP> RUNDOWN -FILE file-name

  MUPIP> RUNDOWN -REGION region-list

MUPIP RUNDOWN Example

- To verify that the memory associated with the file mumps.ubg has been flushed into the database file and the file is closed:

```
MUPIP> RUNDOWN -REG DATA

        %GTM-I-MUFILRNDWNSUC, File successfully rundown.
```

MUPIP FREEZE

- The FREEZE command makes a database unavailable for update
  - Use FREEZE to coordinate database activity with the invocation of an operating system utility that accesses the database file
- The format of the FREEZE command is:

  MUPIP> FREEZE [-qualifier] file-name

  MUPIP> FREEZE [-qualifier] region-list

  - The FREEZE command must include one of the following two qualifiers:
    - ON -- Prohibits updates by “freezing” processes that try to update the file or the region
    - OFF -- Allows updates to the file or the region that has been frozen to continue

MUPIP FREEZE Example

- Use the FREEZE command to stop updates to the DATA region:

```
MUPIP> FREEZE - ON DATA
```

Cache freeze	00000319
Freeze image count	00007A90

  - To verify this, the user can dump the file header for region DATA using DSE and take note of the Cache freeze and Freeze image count values

- Remove the FREEZE command that was just placed on the DATA region:

```
MUPIP> FREEZE -OFF -OVERRIDE DATA
```

Cache freeze	00000000
Freeze image count	00000000

  - Again, to verify this, DSE can be used to dump the file header

MUPIP STOP

- The STOP command terminates a YDB process. This process executes an orderly rundown of all databases in which it currently has interest, and then exits.
  - A MUPIP STOP performs the same as a kill -15.
- The format of the STOP command is:

  MUPIP> STOP process -id

  - The Unix command $ ps -ef|grep mumps may be used to find the process ID.

MUPIP STOP Example

- Find and stop all M processes in the ibstrain directory:

```
$ ps -ef|grep mumps
    johnsonb 31077 31756 0.0 23:56:54 ttyp3    0:00.03 /ydbinst/ydb_dist/mumps -direct
```

— In this case, the process ID found was 31077. To stop that process, execute the following:

```
$ mupip STOP 31077
    STOP issued to process 31077
```

— The user (johnsonb) would receive the following message and be placed back at a UNIX prompt.

```
    YDB>
    Image HALTed by SIGTERM
```

Database Structure Editor (DSE)

The Database Structure Editor is the utility the system uses to analyze and repair databases. Use this document in conjunction with the chapter discussing the Database Structure Editor (DSE) in the YottaDB Administration and Operations Guide.

The DSE utility may be used to:

- dump parts of the database for troubleshooting integrity problems
- add or delete records within a block
- modify block, record, or file header information
- update bitmaps
- save copies of database fragments

Accessing DSE

- After defining the YDB environment, execute from the UNIX prompt:

```
$ dse

DSE>
```

- To move from one database file to another, use the FIND -REGION command to find each database file associated with the corresponding region:

```
DSE> FIND -REGION=DATA

File:  /ydbinst/gbls/mumps.data

Region:  DATA
```

DSE Commands and Qualifiers

To help the new DSE user, the following lists categorize the DSE commands by their potential level of impact on the database

- General Purpose Commands

  Commands do not affect the database

- Non-invasive Commands

  Commands use, but do not change the database

- Invasive Commands

  Commands modify the database and can damage the database or may cause erratic behavior in the application if used incorrectly

- General Purpose Commands
  - —CLOSE
  - —EVALUATE
  - —EXIT
  - —HELP
  - —OPEN
  - —PAGE
  - —SPAWN
- Non-invasive Commands
  - —FIND
  - —DUMP
  - —SAVE
  - —RANGE
  - —INTEGRIT
- Invasive Commands
  - —ADD
  - —ALL
  - —BUFFER_FLUSH
  - —CHANGE
  - —CRITICAL
  - —MAPS
  - —OVERWRITE
  - —SHIFT
  - —REMOVE
  - —RESTORE

DSE File Header

- The DSE File Header contains important information pertaining to the structure of the database
- The database file header is comprised of four sections:
  - Data Elements : used for accounting control and logging purposes.
  - Control Fields : contains fields that are used to show settings for the Access Method used (usually BG).
  - M Lock Space : provides M lock information.
  - Master Bit Maps : contains information describing all local bit maps as being full or not.

DSE File Header (Cont.)

- To view the file header for the UBG region:

```
DSE>FIND -REGION=UBG

File    /ydbinst/gbls/mumps.ubg

Region              UBG

DSE>DUMP -FILEHEADER
```

```
File    /home/jdoe/.yottadb/r1.20_x86_64/g/yottadb.dat
Region  DEFAULT
File            /home/jdoe/.yottadb/r1.20_x86_64/g/yottadb.dat
Region          DEFAULT
Date/Time       27-JAN-2014 03:13:40 [$H = 63214,11620]
  Access method                          MM  Global Buffers               1024
  Reserved Bytes                          0  Block size (in bytes)        1024
  Maximum record size                   256  Starting VBN                  513
  Maximum key size                       64  Total blocks           0x00000065
  Null subscripts                     NEVER  Free blocks            0x0000005E
  Standard Null Collation             FALSE  Free space             0x00000000
  Last Record Backup     0x0000000000000001  Extension Count              100
  Last Database Backup   0x0000000000000001  Number of local maps           1
  Last Bytestream Backup 0x0000000000000001  Lock space             0x00000028
  In critical section            0x00000000  Timers pending                 0
  Cache freeze id                0x00000000  Flush timer           00:00:01:00
  Freeze match                   0x00000000  Flush trigger                960
  Freeze online                       FALSE  Freeze online autorelease    FALSE
  Current transaction    0x0000000000000006  No. of writes/flush            7
  Maximum TN             0xFFFFFFFF83FFFFFF  Certified for Upgrade to       V6
  Maximum TN Warn        0xFFFFFFFD93FFFFFF  Desired DB Format              V6
  Master Bitmap Size                    496  Blocks to Upgrade      0x00000000
  Create in progress                  FALSE  Modified cache blocks           0
  Reference count                         1  Wait Disk                      0
  Journal State                     DISABLED
  Mutex Hard Spin Count                 128  Mutex Sleep Spin Count       128
  Mutex Queue Slots                    1024  KILLs in progress              0
  Replication State                     OFF  Region Seqno    0x0000000000000001
  Zqgblmod Seqno         0x0000000000000000  Zqgblmod Trans  0x0000000000000000
  Endian Format                      LITTLE  Commit Wait Spin Count         16
  Database file encrypted             FALSE  Inst Freeze on Error         FALSE
  Spanning Node Absent                 TRUE  Maximum Key Size Assured      TRUE
  Defer allocation                     TRUE  Spin sleep time mask    0x00000000
  Async IO                              OFF  WIP queue cache blocks          0
  DB is auto-created                  FALSE  DB shares gvstats             TRUE
  LOCK shares DB critical section     FALSE
```

Global Directory Editor (GDE)

- The Global Directory Editor (GDE) is a tool used for creating, examining, and modifying global directories
  - A global directory is a file that identifies:
    - global variables mapped to database files
    - size limits for names and values of globals
    - other database characteristics
  - All M programs that use the same global directory share all the same global variables

GDE Functions

- The main functions of the global directory editor are to:
  - define the mapping of global variables to database files
  - define the character limitations of global variable names and records
  - provide the M Peripheral Interchange Program (MUPIP) with the characteristics (e.g., -ALLOCATION size) used in creating and extending a database file

Accessing GDE

- After defining the environment, enter M.
- From the YDB prompt, execute the following:

```
YDB>D ^GDE

GDE>
```

To leave GDE

- Use the GDE EXIT command to save all changes and return to the shell.

```
GDE> EXIT
```

- Use the GDE QUIT command to discard all changes and return to the shell.

```
GDE> QUIT
```

SHow (Cont.)

```
GDE>SHOW TEMPLATE

                          *** TEMPLATES ***
Region              Def Coll     Rec Size   Key Size  Null Subs   Std Null Coll  Jnl   Inst Freeze on Err  Qdb Rndwn  Epoch Taper  AutoDB  Stats  LOCK Crit
-------------------------------------------------------------------------------------------------------------------------------------------------------
<default>              0           256        64       NEVER           N         N            N              N           Y         N      Y      Sep

Segment                  Active       Acc      Typ    Block        Alloc     Exten   Options
-------------------------------------------------------------------------------------------
<default>                  *          BG       DYN    4096          100       100    GLOB=1024
                                                                                    LOCK=40
                                                                                    RES=0
                                                                                    ENCR=OFF
                                                                                    MSLT=1024
                                                                                    DALL=YES
                                                                                    AIO=OFF
<default>                             MM       DYN    4096          100       100    DEFER
                                                                                    LOCK=40
                                                                                    MSLT=1024
                                                                                    DALL=YES
```

YottaDB Installation

YottaDB (r1.28) Supported OS Version(s)

- Ubuntu 18.04 LTS
- Red Hat Enterprise Linux 7.6
- Debian GNU/Linux 10 (Buster)

YottaDB Traditional Installation

- In this scenario, we will install YottaDB r1.28 for Linux on the training host
- OS : Ubuntu 18.04.02
- YottaDB installation package name is : yottadb_r128_linux_x8664_pro.tgz
  - Download from https://gitlab.com/YottaDB/DB/YDB/-/tags/r1.28
- YottaDB installation directory will be : /ydb/ydb_dist_r128

YottaDB Traditional Installation

- Install required package

```
$ sudo apt-get update
$ sudo apt-get install binutils
```

- Extract YottaDB and Use "root" privilege installation package to /tmp/ydb

```
$ mkdir /tmp/ydb
$ cd /tmp/ydb
$ pwd
/tmp/ydb
$ tar -xzvf /home/ubuntu/yottadb_r128_linux_x8664_pro.tgz
yottadb_r124/plugin/gtmcrypt/Makefile
yottadb_r124/plugin/gtmcrypt/encrypt_sign_db_key.sh
yottadb_r124/plugin/gtmcrypt/gen_keypair.sh
yottadb_r124/plugin/gtmcrypt/gen_sym_hash.sh
yottadb_r124/plugin/gtmcrypt/gen_sym_key.sh
yottadb_r124/plugin/gtmcrypt/gtm_tls_impl.c
yottadb_r124/plugin/gtmcrypt/gtm_tls_impl.h
yottadb_r124/plugin/gtmcrypt/gtm_tls_interface.h
yottadb_r124/plugin/gtmcrypt/gtmcrypt_dbk_ref.c
yottadb_r124/plugin/gtmcrypt/gtmcrypt_dbk_ref.h
yottadb_r124/plugin/gtmcrypt/gtmcrypt_interface.h
          .
          .
          .
          .
```

YottaDB Traditional Installation

- Verify that /ydb/ydb_dist_r128 is empty

```
$ sudo mkdir -p /ydb/ydb_dist_r128
$ ls -l /ydb/ydb_dist_r128
total 0
```

- Run ydbinstall to install YottaDB

```
$ cd /tmp/ydb/yottadb_r128
$ sudo ./ydbinstall --installdir /ydb/ydb_dist_r128 --keep-obj --overwrite-existing
$ ls -l /ydb/ydb_dist_r128
```

YottaDB Environment System Variable

- **ydb_dist** * (gtm_dist) specifies the path to the directory containing the YottaDB system distribution.
- **ydb_gbldir** * (gtmgbldir) specifies the initial value of the $ZGBLDIR ISV. $ZGBLDIR identifies the global directory.
- **ydb_routines** * (gtmroutines) specifies the initial value of the $ZROutines ISV, which specifies where to find object and source code.
- **ydb_log** * (gtm_log) specifies a directory where the gtm_secshr_log file is stored.
- **ydb_prompt** (gtm_prompt) specifies the initial value of the ISV $ZPROMPT, which controls the YottaDB direct mode prompt.
- **ydb_repl_instance** (gtm_repl_instance) specifies the location of the replication instance file when database replication is in use.
- **ydb_repl_instsecondary** (gtm_repl_instsecondary) specifies the name of the replicating instance in the current environment.

ydb_env_set (default)

```
.yottadb
   | -- r
   | -- r1.10
   | | -- g
   | | | -- yottadb.dat
   | | | -- yottadb.gld
   | | ` -- yottadb.mjl
   | | -- o
   | | ` -- utf8
   | ` -- r
   | -- r1.20
    | -- g
    | | -- yottadb.dat
    | | -- yottadb.gld
       | ` -- yottadb.mjl
    | -- o
    | ` -- utf8
     ` -- r
```

```
alias dse="$ydb_dist/dse"
alias gde="$ydb_dist/mumps -run GDE"
alias ydb="$ydb_dist/ydb"
alias lke="$ydb_dist/lke"
alias mupip="$ydb_dist/mupip"
```

YottaDB Installation : Customize Package

- Install required package

```
$ sudo apt-get update
$ sudo apt-get install binutils
```

- Download YottaDB + Octo packages

```
$ cd /tmp
$ git clone https://github.com/mrockstyle/Octo
```

- Install YottaDB + Octo

```
$ cd /
$ sudo tar -xzvf /tmp/Octo/ydb_dist.tar.gz
```

- Setup YottaDB Instance

```
$ cd / $ sudo tar -xzvf /tmp/Octo/ydb_dir.tar.gz
$ cd /ydbdir $ sudo mv yottadb.pc /usr/share/pkgconfig/
```

YottaDB Installation : Customize Package

- YottaDB Binary Path : /ydb/ydb_dist_r128
- YottaDB Instance Path : /ydbdir
- Instance Database File Path : /ydbdir/gbls/

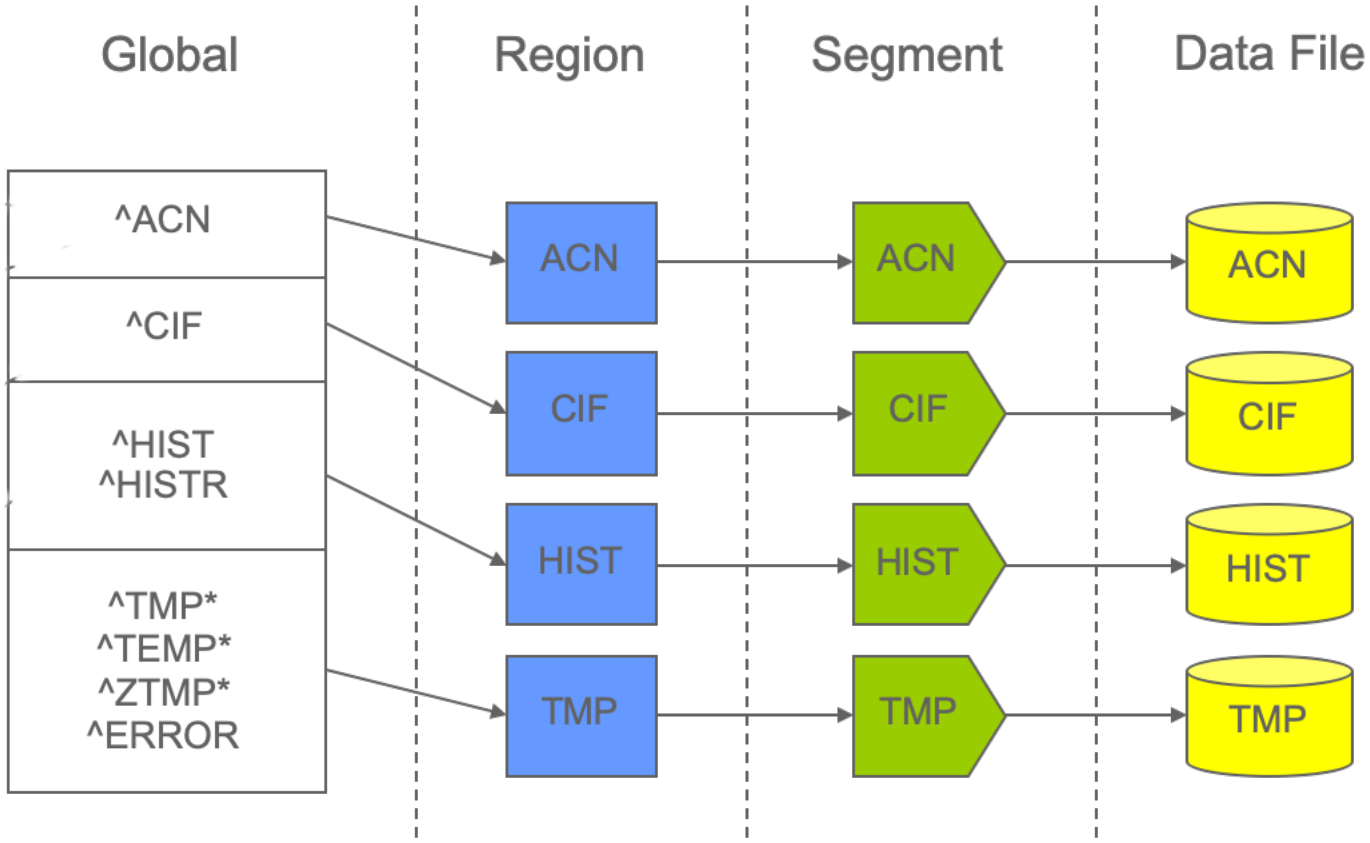
YottaDB DB Structure

Global Directory file

- Store YottaDB DB structure configuration.
- Reference from $ydb_gbldir ($gtmgbldir) system variable.
- View and manage by ”GDE” command

Table / Global / Data File Mapping

- YottaDB provides flexibility in database configuration by allowing each process to distribute its logically monolithic global variable name space across an arbitrary number of database files with the aid of a *global directory* file.



NAME

The NAMES section contains mappings of M global name spaces. More than one name space can map to a single region and a single name space can map to more than one region.

A name space:

- Is case-sensitive
- Must begin with an alphabetic character or a percent sign (%).
- Can be a discrete “global” name, for example, aaa corresponds to the global variable ^aaa.
- Can be a global name ending with a wild card (“*”), for example, abc* represents the set of global nodes which have abc as the starting prefix.
- Can be a subtree of a global name, for example, abc(1) represents a subtree of the global ^abc.
- Can be a subscript range, for example, abc(1:10) represents all nodes starting from ^abc(1) up to (but not including) to ^abc(10).
- A global name can be one to 31 alphanumeric characters. However, the combined length of a global and its subscripts is limited to 1,019 bytes (the maximum key size supported by YottaDB). Note that the byte length of the subscripted global specification can exceed the maximum KeySize specified for its region.
- Maps to only one region in the Global Directory.

REGION

- A region is a logical structure that holds information about a portion of a database, such as key-size and record-size. A key is the internal representation of a global variable name. In this chapter the terms global variable name and key are used interchangeably. A record refers to a key and its data
- A Global Directory must have at least one region. A region only maps to a single segment. More than one name may map to a region
- A region name:
  - Must begin with an alphabetic character, except for $DEFAULT
  - Can include alphanumeric, dollar signs ($), and underscores (_)
  - Can have from 1 to 16 characters
- GDE automatically converts region names to uppercase, and uses DEFAULT for the default region name

SEGMENT

- A segment defines additional database storage characteristics . A segment must map to a single file. A segment can be mapped by only one region
- YDB uses a segment to define a physical file and access method for the database stored in that file
- A segment-name:
  - Must begin with an alphabetic character, except for $DEFAULT
  - Can include alphanumeric, dollar signs ($), and underscores (_)
  - Can have from 1 to 16 characters
- GDE automatically converts segment names to uppercase. GDE uses DEFAULT for the default segment name

FILE

- Files are the structures provided by UNIX for the storage and retrieval of information. Files used by YDB must be random-access files resident on disk
- By default, GDE uses the file-name mumps.dat for the DEFAULT segment. The .dat extension is added to the file name unless an extension is specified

GDE : SHow Command

- The SHOW command displays information contained in the Global Directory about names, regions, and segments
- The format of the SHOW command is:
  - SH[OW]
  - SH[OW] -A[LL]
  - SH[OW] -C[OMMAND] -F[ILE]=[gde-command-file]
  - SH[OW] -N[AME] [name-space]
  - SH[OW] -R[EGION] [region-name]
  - SH[OW] -S[EGMENT] [segment-name]
  - SH[OW] -M[AP] [R[EGION]=region-name]
  - SH[OW] -T[EMPLATE]
  - SH[OW] -A[LL]

GDE Example

- The example below is intended to illustrate how the GDE utility delineates the mapping schema of the example provided above. In this example, we are using two database files, UBG and TBLS. (Actual schema will vary depending upon an individual institution's needs.)

GDE> Show

GDE Example (Result)

```
                                *** TEMPLATES ***
                                                                  Std      Inst
                                        Def      Rec   Key Null     Null     Freeze Qdb   Epoch              LOCK
Region                                 Coll     Size  Size Subs     Coll Jnl on Err Rndwn Taper AutoDB Stats Crit
-------------------------------------------------------------------------------------------------------------------
<default>                                 0      256    64 NEVER     Y    N   N      N     Y     N      Y     Sep

Segment          Active              Acc Typ Block      Alloc Exten Options
-------------------------------------------------------------------------------
<default>          *                 BG  DYN  4096        100   100 GLOB =1024
                                                                     LOCK =  40
                                                                     RES  =   0
                                                                     ENCR = OFF
                                                                     MSLT =1024
                                                                     DALL = YES
                                                                     AIO  = OFF
<default>                            MM  DYN  4096        100   100 DEFER
                                                                     LOCK =  40
                                                                     MSLT =1024
                                                                     DALL = YES

       *** NAMES ***
Global                               Region
-------------------------------------------------------------------------------
*                                    UBG
DBTBL                                TBLS

                                *** REGIONS ***
                                                                                 Std      Inst
                                Dynamic                           Def      Rec   Key Null     Null     Freeze Qdb   Epoch              LOCK
Region                          Segment                          Coll     Size  Size Subs     Coll Jnl on Err Rndwn Taper AutoDB Stats Crit
-------------------------------------------------------------------------------------------------------------------------------------------
TBLS                            TBLS                                0      256    64 NEVER     Y    N   N      N     Y     N      Y     Sep
UBG                             UBG                                 0      256    64 NEVER     Y    N   N      N     Y     N      Y     Sep

                                *** SEGMENTS ***
Segment                         File (def ext: .dat)Acc Typ Block      Alloc Exten Options
-------------------------------------------------------------------------------------------
TBLS                            /ydbinst/gbls/mumps.tbls
                                                    BG  DYN  4096        100   100 GLOB=1024
                                                                                   LOCK=  40
                                                                                   RES =   0
                                                                                   ENCR= OFF
                                                                                   MSLT=1024
                                                                                   DALL= YES
                                                                                   AIO = OFF
UBG                             /ydbinst/gbls/mumps.ubg
                                                    BG  DYN  4096        100   100 GLOB=2048
                                                                                   LOCK=  40
                                                                                   RES =   0
                                                                                   ENCR= OFF
                                                                                   MSLT=1024
                                                                                   DALL= YES
                                                                                   AIO = OFF

                                 *** MAP ***
  - - - - - - - - - - - Names - - - - - - - - - - -
From                            Up to                             Region / Segment / File(def ext: .dat)
-------------------------------------------------------------------------------------------------------------------
%                               DBTBL                             REG = UBG
                                                                  SEG = UBG
                                                                  FILE = /ydbinst/gbls/mumps.ubg
DBTBL                           DBTBL0                            REG = TBLS
                                                                  SEG = TBLS
                                                                  FILE = /ydbinst/gbls/mumps.tbls
DBTBL0                          ...                               REG = UBG
                                                                  SEG = UBG
                                                                  FILE = /ydbinst/gbls/mumps.ubg
LOCAL LOCKS                                                       REG = UBG
                                                                  SEG = UBG
                                                                  FILE = /ydbinst/gbls/mumps.ubg
```

Question and Answer