YottaDB^{TM} Intermediate

Comsan Chanma (Neo)

T.N. Incorporation Ltd.

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.

- YottaDB Refresh Knowledge
- YottaDB Journaling
- YottaDB Journaling Operation
- YottaDB Journaling - Mupip Set
- YottaDB Journaling - Mupip Journal
- YottaDB Recovery from Journal Files
- YottaDB Journal Extract

YottaDB Refreshing Knowledge

YottaDB is Daemon-less

Process that access YottaDB has to perform database processing logic

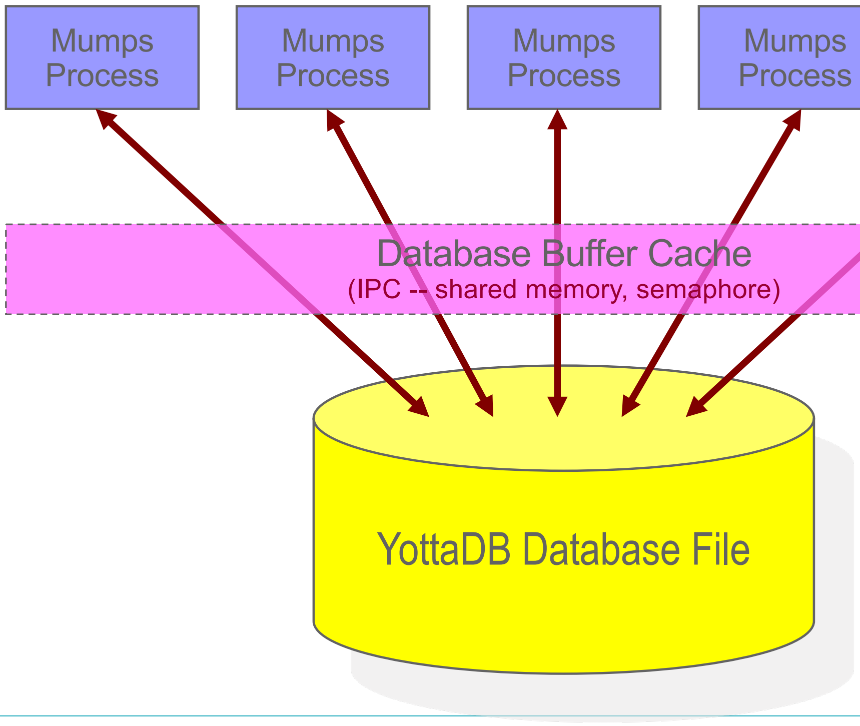
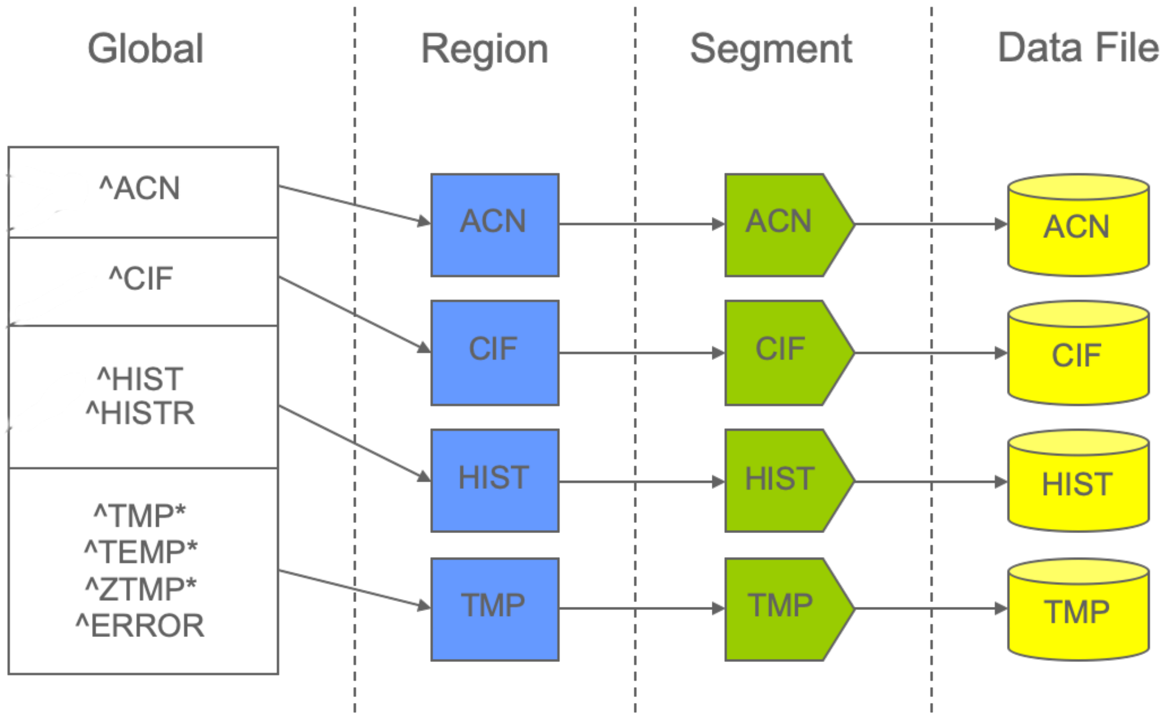


Table / Global / Data File Mapping

- 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.



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.

YottaDB Journaling

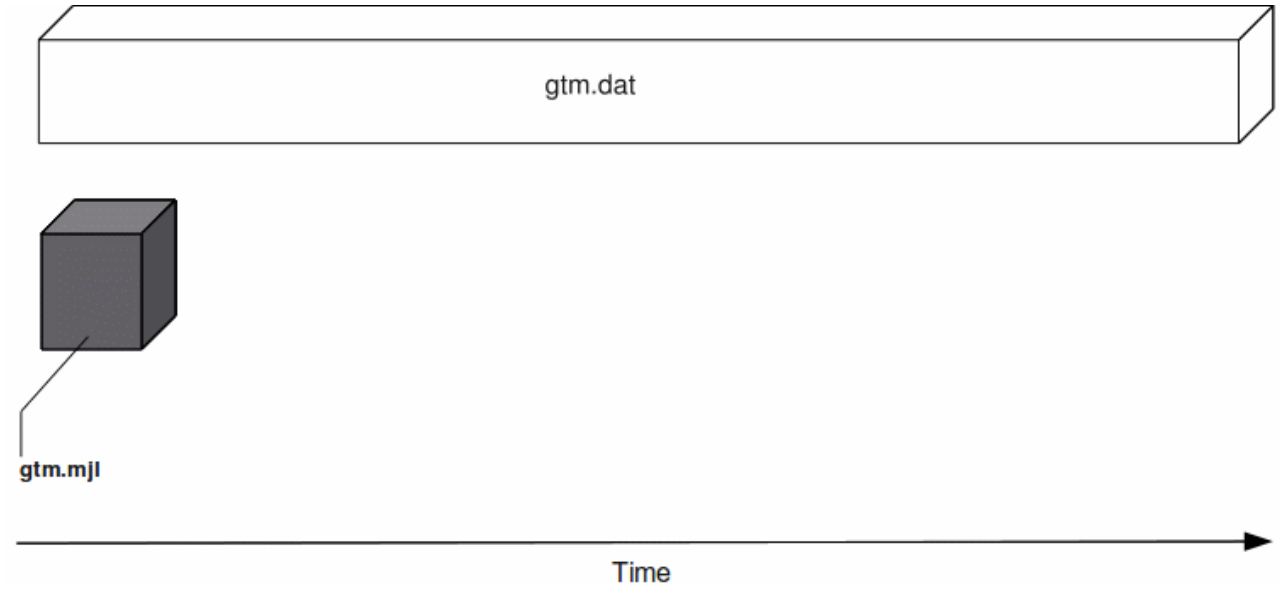
YottaDB Journaling

- YottaDB uses journaling to restore data integrity and provide continuity of business after an unplanned event such as a system crash.
- ACID
  - Atomicity
  - Consistency
  - Isolation
  - **Durability**

YDB Journaling

- YDB use journal files to provide recoverability of databases.
- Journaling provides resiliency against hardware and software failures.
- Journaling uses journal files to record information pertaining to database updates.
- YDB's extensive journaling capabilities are optional.

Journal Files



YYYY - 4-digit-year - such as 2018

JJJ - 3-digit-Julian-day (between 1 and 366) - such as 199

HH - 2-digit-hour in 24 hr format - such as 14

MM - 2-digit minute - such as 40

SS - 2-digit seconds - such as 30

Journaling Benefits

- Automatic "replay" of work to the last committed update recorded in a journal file
- Quick recovery options.
- Recorded database updates formatted appropriately for processing by an M program.
- Identification of processes active when the system failed.

Disadvantage of Journaling

- Journaling requires additional CPU cycles, memory, and disk access.
- Journaling may duplicate features already built into your applications.
- A journal file has questionable value in the case where the database and the journal share a common point of failure that affects the information in both over a significant period of time.

Additional Considerations

- Overhead costs of manual recovery
- The potential risk to the organization from damage to the information during the relatively infrequent and "abnormal" handling of a recovery.
- The cost of reduced computer throughput or, alternatively, the additional hardware to support journaling with the same level of performance.
- Database design.
- The disk space and disk channel overheads associated with journaling.
- Type of data.
- Etc.

Select Database Files for Journaling

- Always journal data that is worth preserving.
- Weigh the deltas associated with manual re-entry and automatic re-play of transactions.
- Journal both frequently updated globals and infrequently updated globals.
- Separate the point of failure for the journal and the associated database files.

Fencing Transactions

- The programming practice of fencing logical transactions protects database integrity during a system interruption.
- A logical transaction is a logical unit that is not complete unless all parts of the transaction are captured.
- the logical transaction "transfer funds between accounts" consists of a debit update to one account and a credit update to another account.

GT.M Fencing Transactions

- TSTART activate transaction fence control.
- The TCOMMIT commands respectively, close the fenced transaction and write the TCOMMIT journal records in all the journal files of all regions involved in the transaction.
- Subsequent SETs and KILLs across all accessed regions are marked as belonging to a fenced transaction.
- If a TSTART has been processed, and another TSTART is encountered prior to a TCOMMIT, increments a "transaction depth" counter accessible as $TLEVEL.
- The TCOMMIT fence is emitted to the journal file when the "transaction depth" returns to zero.
- The maximum depth of a TP transaction nesting is 127 levels.

Fencing Advantages

- Faster recovery
- Minimum risk recovery
- Databases recovered from journals that include fences do not require post-recovery checks and repairs for logical consistency

Fencing Disadvantages

- Must be programmed into the M code
- Fencing requires additional CPU resources and adds entries to the journal file(s)
- Fencing may duplicate methods of recovery already established to address these issues

YottaDB Journaling Operation

YDB Journaling Utility

- The YBD Journaling facility has impact on a number of components:
  - MUPIP SET –JOURNAL
    - used to modify journaling characteristics, and to initiate new journal files.
  - MUPIP JOURNAL
    - used to process journal files for recovery or analysis.
  - DSE
    - used to examine current journaling configuration and historical statistics.

YDB Journaling Utility

- GDE
  - used to set up default journaling characteristics for database regions.
- MUPIP BACKUP
  - used to make database backups - can initiate new journal files.
- MUPIP SET –REPLICATION
  - used to modify replication characteristics - replication requires journaling.

Turning On Journaling

- There are two switches to turn on journaling - ENable/DISable and ON/OFF. i.e. To turn on journaling, use either:

```
mupip set -journal=enable -region '*'
```

or

```
mupip set -journal=on -region '*'
```

- Enabling or disabling journaling requires stand alone access to the database.
- Turning journaling on and off can be done when the database is in use.

- Mupip set with journal properties example

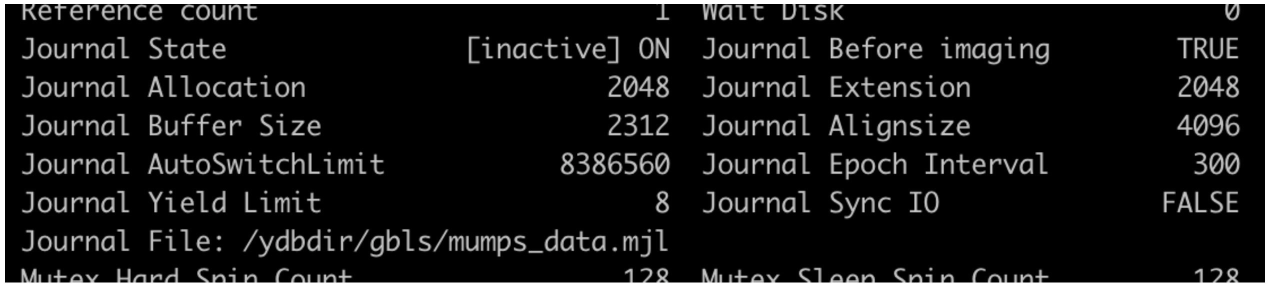
```
mupip set -reg -replication=ON
-journal=on,before,enable,alloc=4194303,exten=8192,auto=8388607,buff=32768,epoch=90,nosync_io “*”
```

Examine Journaling Configuration

- Use DSE utility to dump fileheader

$ dse

DSE> dump -fileheader



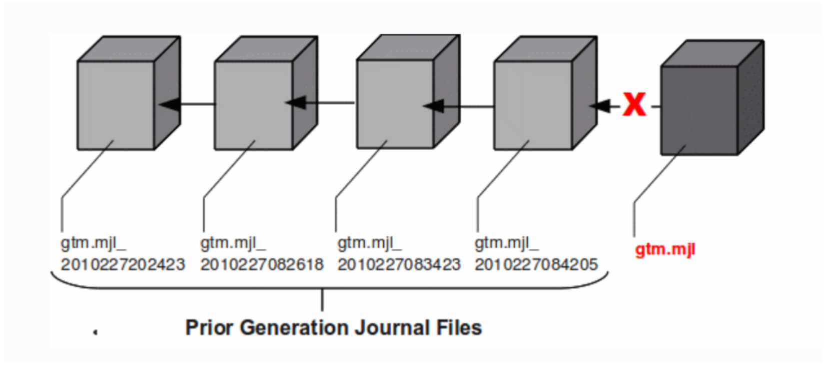
```
Reference count                           1  Wait Disk                          0
Journal State                 [inactive] ON  Journal Before imaging          TRUE
Journal Allocation                     2048  Journal Extension               2048
Journal Buffer Size                    2312  Journal Alignsize               4096
Journal AutoSwitchLimit             8386560  Journal Epoch Interval           300
Journal Yield Limit                       8  Journal Sync IO                FALSE
Journal File: /ydbdir/gbls/mumps_data.mjl
Mutex Hard Spin Count                   128  Mutex Sleep Spin Count           128
```

Processing Journal Files

- MUPIP JOURNAL
  - -EXTRACT : Transfers information from journal files into files formatted for processing by M routines.
  - -RECOVER : Transfers information in the journal file(s) back into database file(s).
  - -SHOW : Reports summary information about journal files.
  - -VERIFY : Reports on journal integrity.

Backup Journal Files

- YottaDB recommends separate backup schemes for database files and journal files.
- Should switch the new journal file at the same point of DB backup.
- MUPIP BACKUP uses the -BKUPDBJNL and -NEWJNLFILES to interact with journal files.
  - BKUPDBJNL enables or turns off the journaling characteristics of the backup database.
  - NEWJNLFILES sets the journaling characteristics of the database being backed up.



MUPIP BACKUP -NEWJNLFILES=NOPREVLINK

Journal Files Housekeeping

- Specify journal files maintain period for recovery.
- Backup and delete inactive journal file that older than the time period.
  - *.mjl_*
  - No any process access the files.

Lab 1.1

1. Dump fileheader to examine current journal configuration.

```
$ . /ydbdir/ydbenv
$ dse
DSE> dump –fileheader
```

2. Turn on journaling of all region and examine the configuration

```
$ mupip set -journal=on -reg “*”
```

3. Turn off journaling of all region and examine the configuration

```
$ mupip set -journal=off -reg “*”
```

4. Disable journaling of all region and examine the configuration

```
$ mupip set -nojournal -reg “*”
```

YottaDB Journaling – Mupip set

MUPIP –SET JOURNAL

- Changes some database characteristics, such as whether journaling is active for a specific file or region(s)

```
SE[T] -qualifier... file-name or region-name
```

MUPIP –SET JOURNAL(Cont.)

- SET Object Identifying Qualifiers

```
-f[ile]

-r[egion]
```

- SET Action Qualifiers

```
-j[ournal][=journal-option-list]

-noj[ournal]
```

MUPIP –SET JOURNAL(Cont.)

- SET –JOURNAL Options
  - disable
    - Equivalent to the –nojournal.
    - No other option are allowed.
  - enable
    - Turn journaling on by default.
    - If turn journaling on, must also specify before_images or nobefore_images.
  - on
    - Only for object with journaling already enabled.
    - Must specify before_images or nobefore_images.
  - off
    - journal characteristics ready to turn ON, but GT.M does not write journal updates to that file.

MUPIP –SET JOURNAL(Cont.)

- SET –JOURNAL Options (Cont.
  - `be[fore_images]`
    - The journal should capture before-images of information that an update is about to modify.
  - `nobe[fore_images]`
    - The journal shouldn’t capture before-images.
  - `f[ilename]=“file-name”`
    - Incompatible with SET –REGION, if more than one region is specified.
    - Journal file name are limited to 255 characters.
    - “mupip create” establishes the journal file name from gld configuration.
    - If gld configuration doesn’t contain the journal file name, “set –journal derives the file name from DB file name “.mjl”
    - Old file is renamed with "_YYYYJJJHHMMSS"

MUPIP –SET JOURNAL(Cont.)

- SET –JOURNAL Options (Cont.)
  - `a[llocation]=blocks`
    - Specifies when GT.M should first review the disk space available for the journal file.
  - `e[xtension]=blocks`
    - Specifies when GT.M should review diskspace available for the journal file after the ALLOCATION has been used up.
  - `bu[ffer_size]=pages`
    - Specifies the amount of memory used to buffer journal file output.
    - By default, MUPIP CREATE establishes the BUFFER_SIZE from the Global Directory, where the default is 128 pages.

MUPIP –SET JOURNAL(Cont.)

- MUPIP SET & Standalone Access to Database
  - the database file transitions from JOURNAL=DISABLED (NOJOURNAL) to JOURNAL=ON or =OFF
  - the database file transitions from JOURNAL=ON or =OFF to JOURNAL=DISABLED (NO JOURNAL)
  - the journal-option-list specifies a BUFFER_SIZE that is different from the current BUFFER_SIZE setting.

Lab 1.2

Run mupip set with various option and examine the result.

```
$ mupip set -file -journal=nobe,buff=128 mumps.data

$ mupip set -region -j=enable,be,a=50000,ext=5000 data

$ mupip set -region -journal=enable,before data,octo

$ mupip set -file -nojournal mumps.data
```

YottaDB Journaling – Mupip Journal

MUPIP JOURNAL

- After using the MUPIP SET command to enable journaling and set characteristics for the file, MUPIP JOURNAL analyzes, extracts from, reports on, and recovers journal files.

```
MUPIP J[OURNAL] -qualifier[...] file-name
```

MUPIP JOURNAL(Cont.)

- Action qualifiers (one or more)
  - `-rec[over]`
    - Instructs the JOURNAL command to replay database updates in the specified journal file into the appropriate database.
    - Performs a STANDALONE check on the database because it needs exclusive access to database files before recovery can occur.
    - Multiple files can be recovered in a single command.
  - `-[no]v[erify]`
    - Checks a journal file for proper form.
    - JOURNAL commands may specify –VERIFY alone or with other action qualifiers.
    - JOURNAL –RECOVER ignores –VERIFY for all qualifier combinations that do not include –FORWARD and –FENCES=NONE

MUPIP JOURNAL(Cont.)

- Action qualifiers (one or more)(Cont.)
  - `-ex[tract][=file-specification]`
    - JOURNAL should transfer the contents of one or more journal files to a single output file in a format intended for processing by an M program.
    - By default, MUPIP JOURNAL derives the output file specification using the name of the journal file and a file type of .mjf.
  - `-sh[ow][=show-option-list]`
    - The show-option-list includes:
      - `AL[L]`
      - `H[EADER]`
      - `P[ROCESSES]`
      - `AC[TIVE_PROCESSES]`
      - `B[ROKEN_TRANSACTIONS]`
      - `S[TATISTICS]`

MUPIP JOURNAL(Cont.)

- Direction qualifiers (one and only one)
  - `-fo[rward]`
    - Specifies that JOURNAL processing should proceed from the beginning of the given journal file.
  - `-ba[ckward]`
    - Specifies that JOURNAL processing should proceed from the end of the journal file.

MUPIP JOURNAL(Cont.)

- Time qualifiers (optional)
  - Absolute format is "day-mm-yyyy hh:mm:ss:cc", indicate today with "- -" before "hh"
  - Delta format is "day hh:mm:ss:cc" .
- `-a[fter]=time`
  - only applies to JOURNAL –EXTRACT –FORWARD
- `-be[fore]=time`
- `-si[nce]=time`
  - specifies a starting time for any action –BACKWARD.
- `-[no]loo[kback_limit][=lookback-option-list]`
  - specifies a "safety zone" for resolving open fenced transactions when processing any action –BACKWARD.
  - the –LOOKBACK_LIMIT= argument may be a list of limits: "TIME=time" and/or "OPERATIONS=integer."

MUPIP JOURNAL(Cont.)

- Control qualifiers (optional)
  - `–red[irect]=file-pair-list`
    - Specifies that JOURNAL –RECOVER replay the journal file to a database different than the one for which it was created.
    - “(old-file-name=new-file-name,…)”
  - `–fe[nces]=fence-option`
    - Specifies how JOURNAL processes fenced transactions.
    - NONE, ALWAYS and PROCESS(default)
  - `–[no]in[teractive]`
    - Specifies whether, for each error over the –ERROR_LIMIT, JOURNAL processing prompts the invoking operator for a response to control continuation of processing.

MUPIP JOURNAL(Cont.)

- Control qualifiers (optional)(Cont.)
  - `–[no]er[ror_limit][=integer]`
    - Specifies the number of errors that JOURNAL processing accepts.
    - By default, MUPIP JOURNAL uses –ERROR_LIMIT=0.
  - `–[no]c[hecktn]`
    - specifies that JOURNAL –FORWARD must verify that the first database update in the journal file has the next transaction number after the current transaction in the database file.
    - By default, JOURNAL –FORWARD uses –CHECKTN.
  - `–[no]d[etail]`
    - specifies that a JOURNAL –EXTRACT includes diagnostic details intended for use by or under the direction of Sanchez Computer Associates.
    - By default, JOURNAL –EXTRACT uses NODETAIL

MUPIP JOURNAL(Cont.)

- Selection qualifiers (optional)
    - Journal Selection Qualifiers are only used with the –EXTRACT operation.
  - `–g[lobal]=global-list`
    - “([-]^global_pattern[*],…)”
  - `–u[ser]=user-list`
    - “([-]user_name[*],…)”
    - By default, JOURNAL processes database updates regardless of the user who initiated them.
  - `–id=pid-list`
    - “([-]pid,…)”
    - By default, JOURNAL processes database updates regardless of the process by which they were initiated.
  - `–t[ransaction]=transaction-type`
    - [-]SET or [-]KILL

Lab 1.3

Run mupip journal with various option and examine the result.

```
$ mupip journal –recover –forward –fences=none mumps_data.mjl
                ------------------------------
$ mupip set -file –journal=before,buff=128 mumps.data
$ mupip set -file –journal=before,buff=128 mumps.octo
…
$ mupip journal –recover –verify –back –error=2 mumps_data.mjl
                ------------------------------
$ mupip jour –forw  -befo=”-- 10:30” -glob=”^TEST*” -extr=bv \
  mumps_data.mjl
                ------------------------------
$ mupip jour –rec –back –befo=”-- 10:30” -since=”-- 9:30” \
  -lookback_limit=”time=000:05” mumps_data.mjl
```

YottaDB Recovery from Journal Files

Recovery from a Journal File

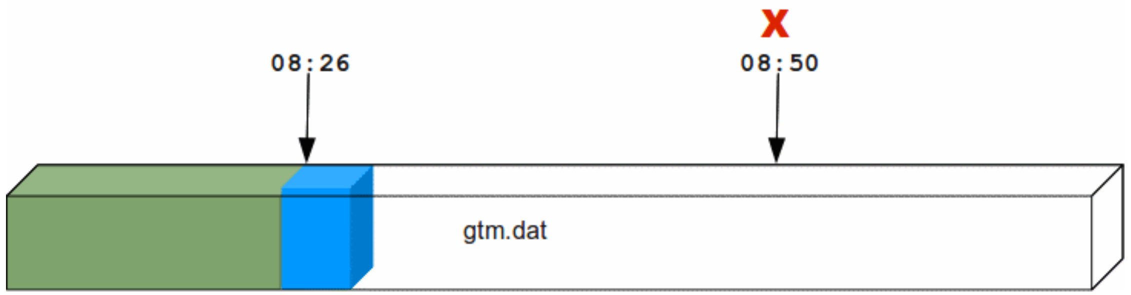
The following two procedures enable recovery of a database from a journal file:

- Forward Recovery (roll forward by applying)
- Backward Recovery (roll back to a checkpoint, optionally followed by a subsequent roll forward)

Forward Recovery

- A Forward Recovery procedure restores a backup copy of the database, and applies the journal file to that database file.
- The starting point for both the database and corresponding journal files must be identical.
- Forward Recovery generally takes longer than Backward Recovery.
- if the current database is destroyed, you must use Forward Recovery.
- If a journal file was created using NOBEFORE_IMAGES, that journal permits only Forward Recovery.

Forward Recovery(Cont.)

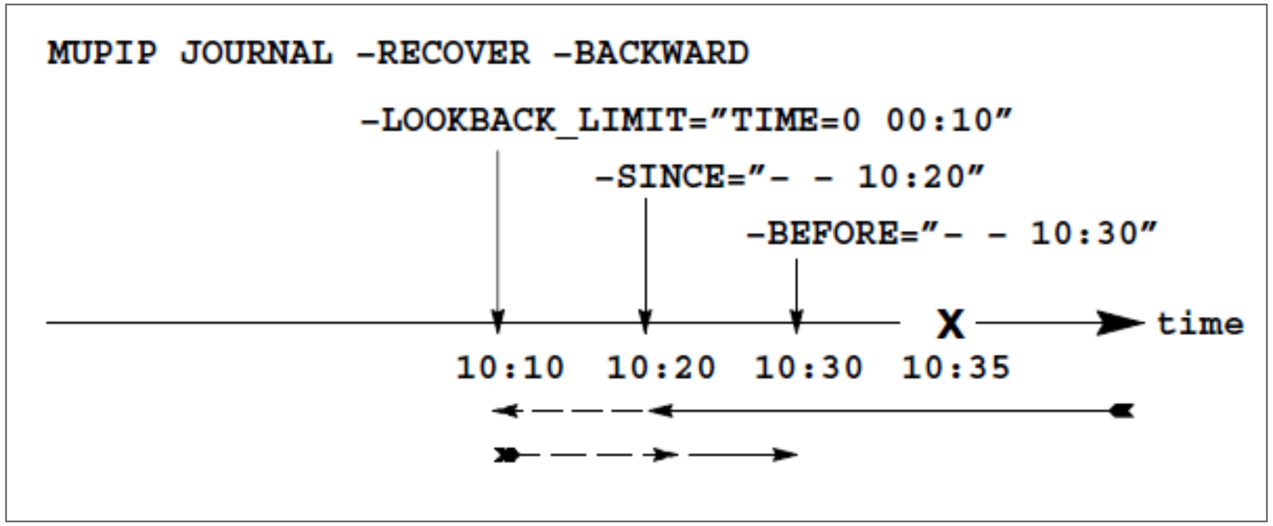


```
mupip journal -recover -forward -before=“- - 8:50” yottadb.mjl
```

Backward Recovery

- Backward Recovery applies the journal file to the active database, moving backward from the end of the file.
- Backward Recovery uses "before-image" journaling.
- Backward Recovery works only if the production database is usable, and if the MUPIP SET command that created the journal file specified the BEFORE_IMAGES qualifier.
- "Before-image" journaling requires more disk I/O and storage space than M-level (or NOBEFORE) journaling.

Backward Recovery(Cont.)

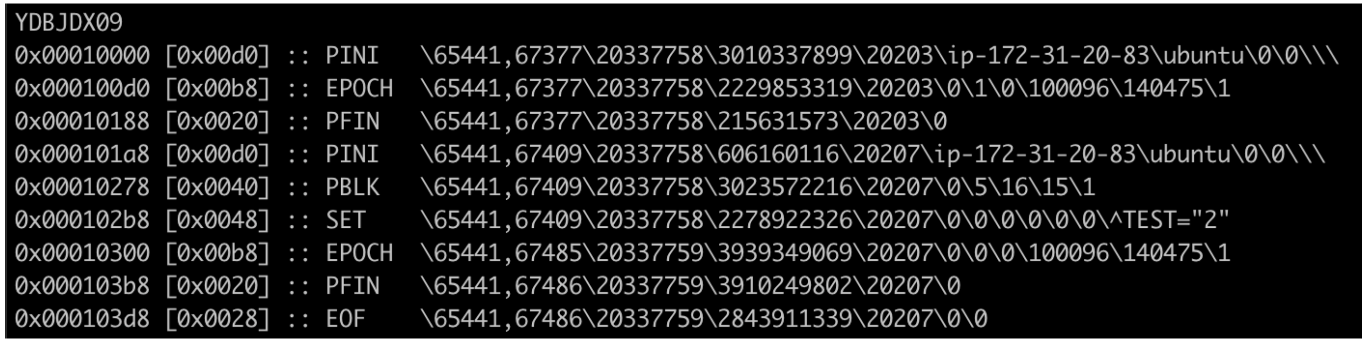

MUPIP JOURNAL -RECOVER -BACKWARD
-LOOKBACK_LIMIT="TIME=0 00:10"
-SINCE="- - 10:20"
-BEFORE="- - 10:30"
X
time
10:10
10:20
10:30
10:35

mupip journal -recover -backward -lookback_limit="TIME=0 00:10" -since="- - 10:20" -before="- - 10:30"

YottaDB Journal Extract

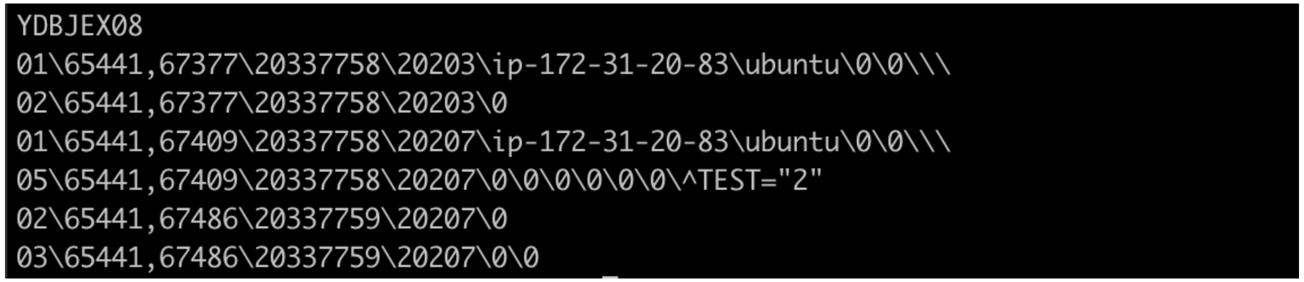
Journal Extract Command

mupip journal -extract -detail -forward mumps_data.mjl



```
YDBJDX09
0x00010000 [0x00d0] :: PINI   \65441,67377\20337758\3010337899\20203\ip-172-31-20-83\ubuntu\0\0\\\
0x000100d0 [0x00b8] :: EPOCH  \65441,67377\20337758\2229853319\20203\0\1\0\100096\140475\1
0x00010188 [0x0020] :: PFIN   \65441,67377\20337758\215631573\20203\0
0x000101a8 [0x00d0] :: PINI   \65441,67409\20337758\606160116\20207\ip-172-31-20-83\ubuntu\0\0\\\
0x00010278 [0x0040] :: PBLK   \65441,67409\20337758\3023572216\20207\0\5\16\15\1
0x000102b8 [0x0048] :: SET    \65441,67409\20337758\2278922326\20207\0\0\0\0\0\0\^TEST="2"
0x00010300 [0x00b8] :: EPOCH  \65441,67485\20337759\3939349069\20207\0\0\0\100096\140475\1
0x000103b8 [0x0020] :: PFIN   \65441,67486\20337759\3910249802\20207\0
0x000103d8 [0x0028] :: EOF    \65441,67486\20337759\2843911339\20207\0\0
```

mupip journal -extract -forward mumps_data.mjl



```
YDBJEX08
01\65441,67377\20337758\20203\ip-172-31-20-83\ubuntu\0\0\\\
02\65441,67377\20337758\20203\0
01\65441,67409\20337758\20207\ip-172-31-20-83\ubuntu\0\0\\\
05\65441,67409\20337758\20207\0\0\0\0\0\0\^TEST="2"
02\65441,67486\20337759\20207\0
03\65441,67486\20337759\20207\0\0
```

Journal Extract Format

NULL	00\time\tnum\pid\clntpid\jsnum\strm_num\strm_seq\salvaged
PINI	01\time\tnum\pid\nnam\unam\term\clntpid\clntnnam\clntunam\clntterm
PFIN	02\time\tnum\pid\clntpid
EOF	03\time\tnum\pid\clntpid\jsnum
KILL	04\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\updnum\nodeflags\node
SET	05\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\updnum\nodeflags\node=sarg
ZTSTART	06\time\tnum\pid\clntpid\token
ZTCOM	07\time\tnum\pid\clntpid\token\partners
TSTART	08\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq
TCOM	09\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\partners\tid
ZKILL	10\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\updnum\nodeflags\node
ZTWORM	11\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\updnum\ztwormhole
ZTRIG	12\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\updnum\nodeflags\node
LGTRIG	13\time\tnum\pid\clntpid\token_seq\strm_num\strm_seq\updnum\trigdefinition

Journal Extract Format

01 record indicates a process/image-initiated update (PINI) into the current journal file for the first time.

02 record indicates a process/image dropped interest (PFIN) in the current journal file.

03 record indicates all YottaDB images dropped interest in this journal file and the journal file was closed normally.

04 record indicates a database update caused by a KILL command.

05 record indicates a database update caused by a SET command.

06 record indicates a ZTSTART command.

07 record indicates a ZTCOMMIT command.

08 record indicates a TSTART command.

09 record indicates a TCOMMIT command.

10 record indicates a database update caused by a ZKILL command.

11 records indicates a value for/from $ZTWORMHOLE (when replication is turned on).

12 record indicates a ZTRIGGER command.

13 record indicates a trigger definition as a logical action from an originating/primary instance to a replicating/secondary instance

Journal Extract Format

- time : full time and date in $HOROLOG format
- tnum : database transaction number
- pid : process ID
- nnam : node name (upto first 20 characters are extracted)
- unam : user name
- term : terminal name
- clntpid : pid of the client (in case of a GT.CM server update), zero (0) otherwise
- clntnnam : node name of the client process (in case of GT.CM)
- clntunam : user name of the client process (in case of GT.CM)
- clntterm : terminal name of the client process (in case of GT.CM)

Question and Answer