SWC DB: Abstract

author: Kashirin Alex
kashirin.alex@gmail.com

SWC DB

(Super Wide Column Database)

https://github.com/kashirin-alex/swc-db

The major differences “Super Wide Column Database” has to commonly known Wide Column Databases are SWC-DB does not have Tables nor Namespaces and while cell key as known to be in Wide Column Database structured in timestamp, row, column-family and column-family-qualifier in SWC-DB a cell key is a list of Fractions with timestamp. The differences in SQL structure, it is in-place of '*select columns from “table_name”*;' with SWC-DB It is ' *select [where_clause [Columns-Intervals]];* '. Considering to structure a Wide-Column-DB in SWC-DB it can be in these forms key=[F(row), F(column-family), F(column-family-qualifier)] or the actual column is column-family with key=[F(row), F(column-family-qualifier)].

The Fractions in SWC-DB cell-key let numerous “qualifiers”, as known to be, with a range-locator able to respond with the ranges applicable to the fractions of a scan specs. As a result a scan-spec of key=[>“”, >=”THIS”] will scan ranges that consist the “THIS” on comparator with a help of meta-column that include, additionally to the key-begin and key-end of a range, the minimal and maximum values of the fractions in an aligned manner. Hence the name “Super Wide Column“ a column can have cells with one key [F(1st)] second key [F(1st), F(2nd)] third key [F(1st), F(2nd), F(3rd), ..] and the scan/select is possible on [F(1st)] and above that will return all the cells having fraction one equal “1st ” and so as without further indexations to select cells with key [>F(), F(2nd)] returning the cells with second fraction equal “2nd”.

The comparators available in SWC-DB are NONE, PF ,GT, GE, EQ, LE, LT, NE, RE while some have limitations for range-locator as regexp is evaluated as NONE being anything-match. Additionally the conditions of comparators applied on the corresponding “ key-sequence ” by column's schema that include LEXIC, VOLUME, FC_LEXIC, FC_VOLUME that define the sequence of cells in a range. If a prefix (PF) is desired than the choice will be the LEXIC or with FC_LEXIC as VOLUME (volumetric) will not correspond to the char-byte sequence while if desired to have for example a decimal sequence of 0, 1, 2 .. 11 the VOLUME is the right choice whereas the FC_VOLUME unlike tree-wise on fraction keeps the sequence of smaller key fractions-count at the beginning in the range.

SWC-DB use a self-explanatory master-ranges that define ranges to meta-ranges of data-ranges(cells-range) whereas on range-locator scan includes the Key comparison on the comparators of request, resulting in most narrowed ranges for the scan of cells. For the purpose SWC-DB have reserved columns 1: Ranges("SYS_MASTER_LEXIC"), 2: Ranges("SYS_MASTER_VOLUME"), 3: Ranges("SYS_MASTER_FC_LEXIC"), 4: Ranges("SYS_MASTER_FC_VOLUME"), 5: Ranges("SYS_META_LEXIC"), 6: Ranges("SYS_META_VOLUME"), 7: Ranges("SYS_META_FC_LEXIC"), 8: Ranges("SYS_META_FC_VOLUME"), 9: Statistics("SYS_STATS"). The Statistics column used for internal systems monitoring and it can be used like any other counter column (keeping for the purpose) with fraction of [period, role, instance, metric] with counter value

The storage-form in the SWC-DB on FS is based by column-id and range-id, that on path consist CellStores and CommitLog files while at any point one server is responsible for a range-id on column-id and of a path root. The CellStores are files storing Cells in serialized form that are after latest compaction whereas Commit-Log is Fragments of current added data, one fragment is written at a time on a threshold reach or on shutdown.

The limitations that can be over-seen are:

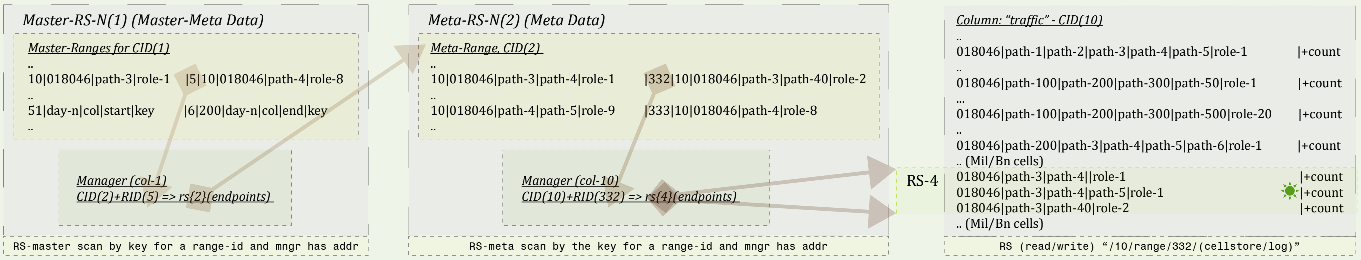
- ✗ Maximum number of columns, it is store-size of int64(2^{64}) – 10(reserved cols) which can be improved by CID to be a string-type.
- ✗ Maximum size of Value or Key-Fraction(after serialization), it is 4GB, while for such data size other limitations apply.

The capabilities to expect:

- ✔ A Manager-Root with definitions of 1K^{2} ranges (a use of 1 GB RAM) is a definition of 1K^{4} Meta-Ranges that sums-down to 1K^{8} Data-Ranges, with range-size configuration to 10GB that makes a total storage volume for a cell size average of 256KB to be a quarter of Yotta Byte.
- ✔ A client can read at 100%(while Client's and Ranger's are equivalent) bandwidth, considering a perfect scan case of each client is requesting on different ranges, number of clients at a given time can be by the number of data Rangers using 100% bandwidth each.
- ✔ Maximum number of concurrent connections to a given server instance, it is the total available ports on the server by the number of configured IPv4 and IPv6 with support of multi-homed / multiple interfaces,

Some examples:

- Search indexing at https://thither.direct/opensearch/ with Wide Column it is being row=”sequences-of-words:domain:path” cf=”lang” whereas with Super Wide Column it can be changed to key=[“sequences-of-words”, “domain”, “path”, “lang”], makes the scan-select much optimized, especially if to query words-data of a domain & path, it would go on to ranges that start with domain & path skipping the seek through ranges of several other many domains that as well include the same word-sequences. While to have the same query on a Wide Column would require tripling the volume of data by using more indexes of word-sequences on a domain (and path) such as. row=”domain:sequences:path” & row=”domain:path:sequences”. At current period the “open-search” on Thither.Direct does not offer querying data(words) on a site:domain or info:url-path as it is unreasonable over the data-volume overheads.
- A theoretical requirement for a building security tracking. Track of how many(an atomic-counter) personnel passed in an area of a building by role on a day:



Master-RS-N(1) (Master-Meta Data)

Master-Ranges for CID(1)
..
10|018046|path-3|role-1 |5|10|018046|path-4|role-8
..
51|day-n|col|start|key |6|200|day-n|col|end|key
..

Manager (col-1)
CID(2)+RID(5) => rs{2}(endpoints)

RS-master scan by key for a range-id and mngr has addr

Meta-RS-N(2) (Meta Data)

Meta-Range, CID(2)
..
10|018046|path-3|path-4|role-1 |332|10|018046|path-3|path-40|role-2
..
10|018046|path-4|path-5|role-9 |333|10|018046|path-4|role-8
..

Manager (col-10)
CID(10)+RID(332) => rs{4}(endpoints)

RS-meta scan by the key for a range-id and mngr has addr

RS-4

Column: “traffic” - CID(10)
..
018046|path-1|path-2|path-3|path-4|path-5|role-1 |+count
..
018046|path-100|path-200|path-300|path-50|role-1 |+count
...
018046|path-100|path-200|path-300|path-500|role-20 |+count
..
018046|path-200|path-3|path-4|path-5|path-6|role-1 |+count
.. (Mil/Bn cells)
018046|path-3|path-4||role-1 |+count
018046|path-3|path-4|path-5|role-1 |+count
018046|path-3|path-40|role-2 |+count
.. (Mil/Bn cells)

RS (read/write) “/10/range/332/(cellstore/log)”

SWC DB: Data topology

author: Kashirin Alex
kashirin.alex@gmail.com

Configuration Settings:

swc.rgr.Range **(defaults)**

.CellStore.count.max=10 , .CellStore.size.max=1GB , .block.size=64MB, .block.encoding=snappy, .compaction.size.percent=33

Ranger (#):

Column (#):

Range (#), (Key Begin <= interval <= Key End):

Range Definer (range.data):

➔ **Header**: (13-byte)
Version(i8), Data-Length(i32), Data-Checksum(i32), Header-Checksum(i32)

➔ **Data**:
CellStores-count(i32), [CellStore-ID(i32), Key-Interval Begin + End]

Commit Log, Fragment(#):

➔ **Header**: (7-byte)
Version(i8), Extension-Length(i32), Checksum(i32)

➔ **Extension**:
Interval, Encoder(i8), Enc-Data-Length(i32), Data-Length(i32), Cells-count(i32)

➔ **Data**:
Cells [Flag(i8) | Fractions-count(i8) | [Fraction(length(i24) | data)] | Control(i8) | Timestamp(i64) | Revision(i64) | Value-length(i32) | Value-Data]

Fragment (#) ++ >= (swc.rs.Range.CommitLog.*roll.size*)

CellStore (#), (Key Begin <= interval < Key End):

Block (#), (Key Begin <= interval < Key End):

➔ **Header**: (17-byte)
Encoder, Data-Enc-Length, Data-Length, Cells-count, Checksum

➔ **Data**:
Cells (serialized)

. .

Block (#) ++ <= (swc.rgr.Range.CellStore.size.max / .DefaultBlockSize)

Blocks Index:

➔ Compressor:	Type			
➔ Uncompressed:	Length			
➔ Checksum: value				
➔ Intervals:	count			
➔ Key Intervals:				
	Begin()	-	End(k1,k5,k5,k5) :	CS position offset
	Begin(k5,k5,k5,k5)	-	End(k1,k7,k7,k7) :	CS position offset

Trailer:

➔ CS Key Interval:	Begin + End
➔ Blocks-Index:	CS position offset
➔ Blocks-Index size:	Length
➔ Trailer-Start:	CS position offset
➔ CS Version:	Value(1)

. .

CellStore (#) ++ <= (swc.rgr.Range.CellStore.count.max)

. .

Range (#) ++

. .

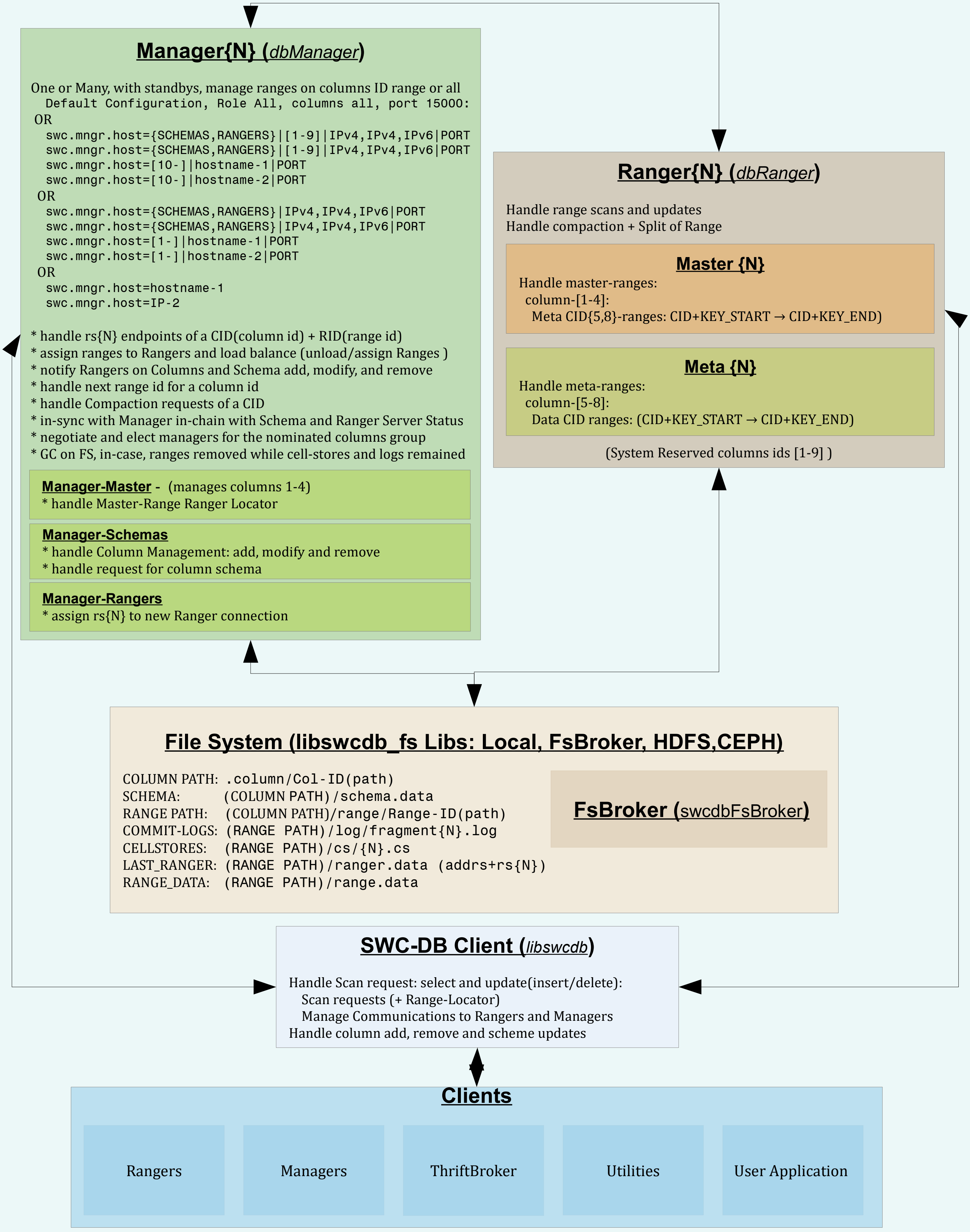
Column (#) ++

. .

Ranger (#) ++

SWC DB: Applications

author: Kashirin Alex
kashirin.alex@gmail.com



Manager{N} (*dbManager*)

One or Many, with standbys, manage ranges on columns ID range or all

```
  Default Configuration, Role All, columns all, port 15000:
 OR
  swc.mngr.host={SCHEMAS,RANGERS}|[1-9]|IPv4,IPv4,IPv6|PORT
  swc.mngr.host={SCHEMAS,RANGERS}|[1-9]|IPv4,IPv4,IPv6|PORT
  swc.mngr.host=[10-]|hostname-1|PORT
  swc.mngr.host=[10-]|hostname-2|PORT
 OR
  swc.mngr.host={SCHEMAS,RANGERS}|IPv4,IPv4,IPv6|PORT
  swc.mngr.host={SCHEMAS,RANGERS}|IPv4,IPv4,IPv6|PORT
  swc.mngr.host=[1-]|hostname-1|PORT
  swc.mngr.host=[1-]|hostname-2|PORT
 OR
  swc.mngr.host=hostname-1
  swc.mngr.host=IP-2
```

* handle rs{N} endpoints of a CID(column id) + RID(range id)
* assign ranges to Rangers and load balance (unload/assign Ranges)
* notify Rangers on Columns and Schema add, modify, and remove
* handle next range id for a column id
* handle Compaction requests of a CID
* in-sync with Manager in-chain with Schema and Ranger Server Status
* negotiate and elect managers for the nominated columns group
* GC on FS, in-case, ranges removed while cell-stores and logs remained

Manager-Master - (manages columns 1-4)
* handle Master-Range Ranger Locator

Manager-Schemas
* handle Column Management: add, modify and remove
* handle request for column schema

Manager-Rangers
* assign rs{N} to new Ranger connection

Ranger{N} (*dbRanger*)

Handle range scans and updates
Handle compaction + Split of Range

Master {N}

Handle master-ranges:
 column-[1-4]:
  Meta CID{5,8}-ranges: CID+KEY_START → CID+KEY_END)

Meta {N}

Handle meta-ranges:
 column-[5-8]:
  Data CID ranges: (CID+KEY_START → CID+KEY_END)

(System Reserved columns ids [1-9])

File System (libswcdb_fs Libs: Local, FsBroker, HDFS,CEPH)

```
COLUMN PATH:  .column/Col-ID(path)
SCHEMA:       (COLUMN PATH)/schema.data
RANGE PATH:   (COLUMN PATH)/range/Range-ID(path)
COMMIT-LOGS:  (RANGE PATH)/log/fragment{N}.log
CELLSTORES:   (RANGE PATH)/cs/{N}.cs
LAST_RANGER:  (RANGE PATH)/ranger.data (addrs+rs{N})
RANGE_DATA:   (RANGE PATH)/range.data
```

FsBroker (swcdbFsBroker)

SWC-DB Client (*libswcdb*)

Handle Scan request: select and update(insert/delete):
  Scan requests (+ Range-Locator)
  Manage Communications to Rangers and Managers
Handle column add, remove and scheme updates

Clients

Rangers
Managers
ThriftBroker
Utilities
User Application

September 16, 2020

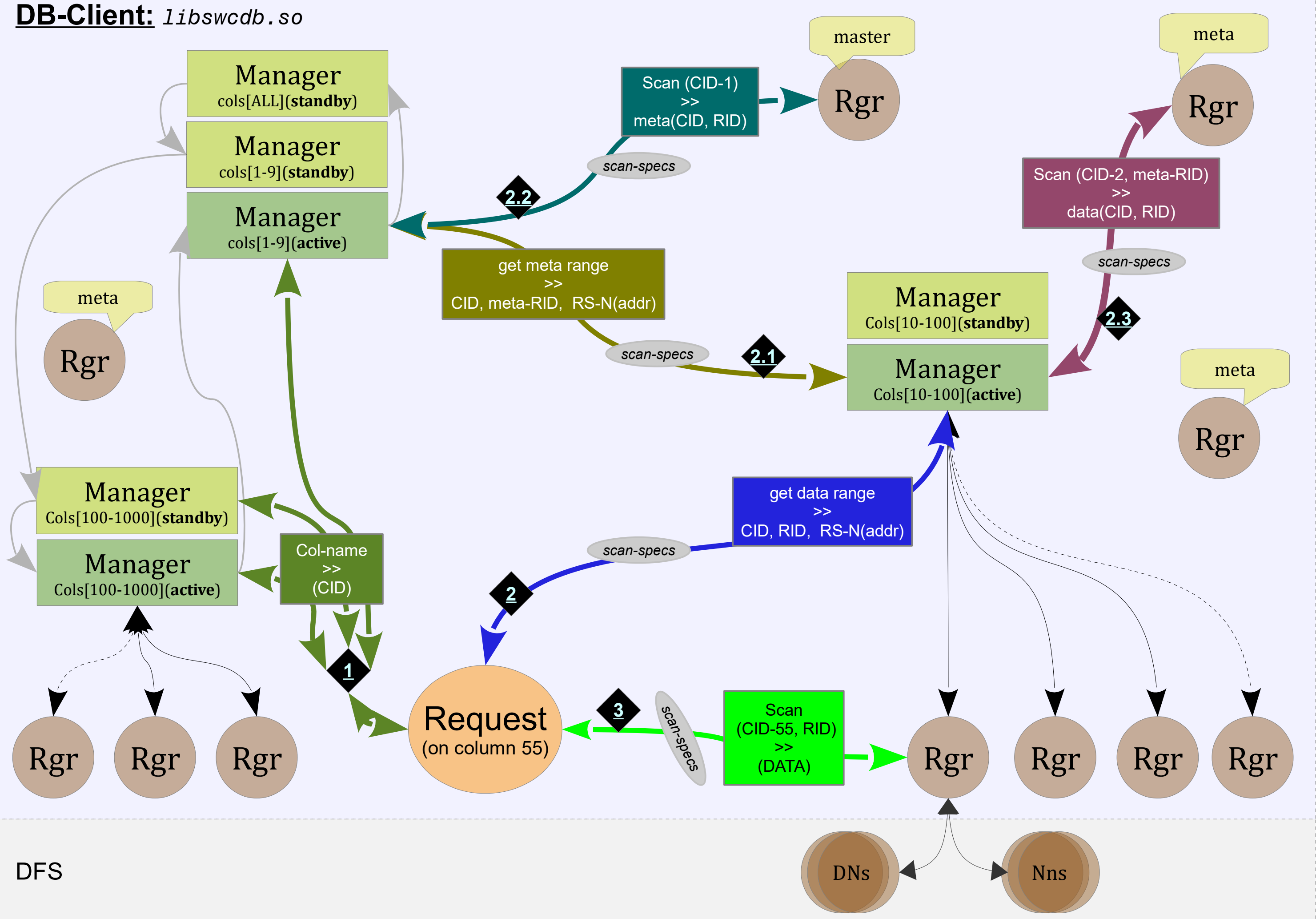
SWC DB: Failure Tolerance

author: Kashirin Alex
kashirin.alex@gmail.com

- ✔ A failed request to a Manager is a connection fail-over to next in list from 'swc.mngr.host' configuration.
- ✔ A failed request to a Ranger(Master, Meta, Data)-N is fail-over to the new newly assigned Ranger(addr) by Manager.
- ✔ Manager, on interval or shut-down state of a managed Ranger(either role), request to load ranges to another Ranger.
- ✔ Distribute File System, depends on the system and it's feature of routing to a datanode .
- ✔ Managers or Rangers in case of a connection or file-descriptor failure try to reconnect to the DFS.
- ✔ Communications security, SSL applicable between servers for non-secure networks.
- ✔ Communication over-heads of resolved-data of column-name, RID-location and Ranger-address are kept on TTL/KA.

In worst case of outdated data being used with a request the Ranger return an error of a NOT_LOADED_RANGE.

DB-Client: `libswcdb.so`



SWC DB: LIB-DB, Scan request (+ Range-Locator)

author: Kashirin Alex
kashirin.alex@gmail.com

Basic Process Flow of Scan request (+ Range-Locator)

Ranges Scan is done on per column base in-parallel(a client's max-range-locators config) with column's Scan Specifications

```
Scan-Specifications = cid, ScanSpecCellKey(key_start, key_finish)
result = new_results = 0,
last_cell_key = rid(meta) = 0,
key_start(meta,data) = ScanSpec.key_start
```

```
1       get column-ID by name => (cid)
        Req. RS-MANAGER[cid-1](SCHEME-MNGR) – req, ([=”ReqColName”]) => (cid)
```

```
DO get_ranges_data:
2       get range-data by (cid, key_start(meta,data), key_finish, rid(meta))
                 => (cid,rid(data),rs{N}(addr), next_rid(meta,data)?, rid(meta))
        Req. RS-MANAGER[cid]:
                If not rid(meta):
2.1                     get range-meta by (cid, key_start(meta), key_finish)
                                => (cid, rid(meta), rs-meta{N}(addr), next_rid(meta)?):
                        Req. RS-MANAGER[cid-1]:
2.2                             get range-master-meta => (rid(meta), next_rid(meta)?)
                                Req. RS-MASTER:
                                        Scan-do (2-cell)(cid-1, [>=”cid”, key_start(meta)], [<”cid”, key_finish]) = rid(meta)
2.3             get range-data by (cid, rid(meta), key_start(data), key_finish)  => (cid, rid, rs{N}(addr), next_rid(data)?):
                Req. RS-META - rs-meta{N}(addr):
                        Scan-do (2-cell)(cid-2, [>=”cid”, key_start(data)], [<”cid”, key_finish]) = rid(data)

        If no range-data:
                goto finish
EXCEPT COMM:
        goto get_ranges_data
```

```
DO scan_range_data:

3       scan range-data by (cid, rid(data), ScanSpecs)  => (new_results):
        Req. RS-DATA – rs{N}(addr):
                Scan-do (cell-limit) (ScanSpecs) = results(data)

        if new_results
                (call_back) (available results), result+=new_results
                last_cell_key=more_results[-1]

EXCEPT COMM, NOT_LOADED_RANGE:
        goto get_ranges_data
```

```
if result < limit(cell_limit):
        # Move Scan Offset by key_start changed to last_cell_key, setting -ge comparator to -gt
        if next_rid(data):
                start_key(data) = last_cell_key
                goto get_ranges_data

        if next_rid(meta):
                rid(meta) = 0
                start_key(meta) = last_cell_key
                goto get_ranges_data

DO finish:
        return result (call_back)
```

SWC DB: Query (SQL) scan

author: Kashirin Alex
kashirin.alex@gmail.com

```
select [where_clause [Columns-Intervals or Cells-Intervals]] [Flags(global-scope)];
```

Columns-Intervals: *if not set, it is all columns from key start to finish.*

```
col(column-name-a1) = ( [Cells-Intervals] [and] [Cells-Intervals] [and] .. [Cells-Intervals] )
[and] ..
col(column-name-b1, .., column-name-b2) = ( [Cells-Intervals] [and] [Cells-Intervals] [and] .. [Cells-Intervals] )
```

Cells-Intervals: *if not set, it is key start to finish.*

```
 cells = ( [Cells-Interval] Flags(interval-scope) )
 [and]..
 cells = ( [Cells-Interval] Flags(interval-scope) )
```

Cells-Interval:

```
[ Condition-Range ] [and] [ Condition-Key ] [and] [ Condition-Value ] [and] [ Condition-Timestamp ]
```

Condition-Range: The applicable ranges for a scan, comparators are always -ge or -le

```
Cell::Key [ <= ]  range  [ <= ] Cell::Key
```

Condition-Key: *key comparator apply to every fraction that do not have a dedicated comparator, exact-match is* key=('k1', 'k2',,,'kN')

```
Key [<|<=|>|>=|=] [ [comparator] "str-1", [comparator] "str-2", [comparator] "str-3", [comparator] "str-N"]

      or (an interval)

[[comparator] "str-1",  [comparator] "str-N"]       [ <= ]  key  [ <=  ]       [[comparator] "str-1",  [comparator] "str-N"]
```

Condition-Value:

```
value [comparator(extended logic options: GE,LE,GT,LT are LEXIC and with 'V' VOLUME as -VGE)] "string"

      or (for columns of counter type), not applicable comparators (prefix and regexp)

value [comparator] "int64_t(string)"
```

Condition-Timestamp: *not applicable comparators (prefix and regexp)*

```
timestamp [comparator] "YYYY/MM/DD HH:MM:ss.mmmuuunnn"

      or (an interval)

"YYYY/MM/DD HH:MM:ss.mmmuuunnn"  [ <=  or  < ]  timestamp  [ <=  or  < ]  "YYYY/MM/DD HH:MM:ss.mmmuuunnn"
```

Comparator:

```
[ =^ ]  :   prefix   (starts-with)
[ >  ]  :   -gt      (greater-than)
[ >= ]  :   -ge      (greater-equal)
[ =  ]  :   -eq      (equal)
[ <= ]  :   -le      (lower-equal)
[ <  ]  :   -lt      (lower-than)
[ != ]  :   -ne      (not-equal)
[ re ]  :   regexp   (regular-expression)
```

Flags: *at global-scope apply to Cells-Interval flags to which does not have flags definitions*

```
[ only_keys ]      =  TRUE on set            # default FALSE
[ only_deletes ]   =  TRUE on set            # default FALSE
[ limit            =  NUMBER(uint32_t) ]     # default ALL
[ limit_by         =  "KEYS" or ".." ]       # default KEYS
[ offset           =  NUMBER(uint32_t) ]     # default 0
[ offset_by        =  "KEYS" or ".." ]       # default KEYS
[ max_versions     =  NUMBER(uint32_t) ]     # default ALL
```

An Example:

```
select
 where
  col(ColNameA1) = (
   cells = ( range >= ['1-'] and ( [>='1-'] <= key = [<='1-1-',="1"] and value = "Value-Data-1" and timestamp > "2010/05/29" limit=10 limit_by="KEYS" )
  )
  and
  col(ColNameB1, ColNameB2) = (
   cells = ( [>='2-'] <= key = [<='2-2-',"1"] and value = "Value-Data-2" and timestamp > "2010/05/29" )
   and
   cells = ( key = [<='21-',"1"] and timestamp > "2010/05/29" )
  )
  max_versions=1;
```

SWC DB: Scan Specs & Results

author: Kashirin Alex
kashirin.alex@gmail.com

Scan Specs, lib-DB-Client:

```
SpecsScan (

  Columns   columns;
  Flags     flags;

)
```

```
SpecsColumn (

  int64_t    cid;
  Intervals  intervals;

)

The object-type is applied to the range-locator (Client)
```

```
SpecsInterval (

  Cell::Key  range_begin, range_end;
  Key        key_start, key_finish;
  Value      value;
  Timestamp  ts_start, ts_finish;
  Flags      flags;

  Cell::Key  offset_key;
  int64_t    offset_rev;
)
```

```
SpecsKey (

  uint32_t  count;
  uint32_t  size;
  uint8_t*  data(serial);

)
```

```
SpecsTimestamp (

  int64_t          value;
  Condition::Comp  comp;

)
```

```
SpecsValue (

  uint8_t*         data;
  uint32_t         size;
  Condition::Comp  comp;

)
```

```
SpecsFlags (


  uint32_t   limit, offset, max_versions;
  uint8_t    options;

)
```

Scan Response, lib-DB-Client:

```
Result (

    List<Col>   cols
    // ResponseFlag status = OK/PARTIAL/ERROR
    // Strings error_rs = [“N”,]

)
```

```
Col (

    String      name
    String      id
    List<Cell>  cells

)
```

```
Cell (

    list<c-array>   key
    int64_t         timestamp
    c-array         value
    uint32_t        value_len

)
```

SWC DB: Column Schema & Actions on Columns

author: Kashirin Alex
kashirin.alex@gmail.com

Although, there are schemas in the SWC-DB these can be considered as schema-less, exception to TTL, Counter and Max-Versions at the Cells level.

Configuration Options:

The following configurations available in the Column-Schema:

- COLUMN-LEVEL:
  - NAME: (string) - column-name
  - CID: (uint64_t) - column-id
  - TYPE: (enum) - plain/counter_i8/counter_i16/counter_i32/counter_i64, default - PLAIN
- CELL-LEVEL:
  - TTL: (uint32_t) – Time To Live in seconds
  - MAX_VERSIONS: (uint32_t) - default 1 - not applicable with COUNTER
- BLOCK-LEVEL:
  - ENCODING: (enum) - none/snappy/zlib (..zstd/bmz/lzo/quicklz)
  - BLOCKSIZE: (uint32_t) - size of a block
  - CELLS: (uint32_t) - number of cells in a block
- CELLSSTORE-LEVEL:
  - REPLICATION: (int8_t) – replication factor applied to the DFS supporting file-replication , default 3
  - SIZE: (int32_t) – max allowed cellstore size in bytes
- RANGE-LEVEL:
  - CS-MAX: (int8_t) – number of cellstores allowed in a range before range-split
  - COMPACT-%: (int8_t) – relative percentage to cellstores-volume allowed without compaction