StoreFile

SortedList<KeyValue>

Sort(row,family,type,column,ts)

RegionUniqueID = Creation Stamp

Represents Puts and Deletes that occurred over a Range of Time:

[ID, IDofNextOldestStoreFile)

Generic Outline of Processing a Get Request

Instantiate server-side GetXServer object

GetXServer(GetX, DeleteSet, Filter)

Seek to start of GetXServer.getRow()

Iterate down KeyValues

For each KeyValue “cur”,

GetXServer.compareTo(cur)

Handle according to below

Get Request

All GetX implement Get and Get.compareTo(KeyValue)

GetServer.compareTo(cur) returns

`-1` if cur should not be included in result
`0` if cur should be included in result
`1` if cur is past this StoreFiles boundary for this Get
`2` if cur is past this Stores boundary for this Get

GetServer.compareTo(cur) implementation

if nothing left to match on this StoreFile, return 1
if nothing left to match on this Store, return 2
compare row
    if next row, return 1
compare family
    if next family (in the future), return -1
compare type
    if type != Put, add to DeleteSet, return -1
compare column+timestamp against Get qualifiers
    if does not match, return -1
    if matches, DeleteSet.compareTo(cur)
        if -1, return -1
        if 0, Filter.match(cur)
            if True, return 0
            if False, return -1
        if 1, return 1

Server-side Handling of GetServer.compareTo() result

`-1` look to the next KeyValue
`0` add cur to ResultSet<KeyValue>
`1` go to next StoreFile
`2` stop, return current ResultSet (early-out)

DeleteSet

SortedList<KeyValue>

DeleteSet.compareTo(KeyValue)

Represents the Deletes that have been read from previous StoreFiles and the Memcache

DeleteSet.compareTo(cur)

`-1` if cur has been deleted
`0` if cur has not been deleted
`1` if cur and everything below it have been deleted

DeleteSet.addDeletes(List<KeyValue>)

Merges existing DeleteSet with deletes in specified list
Removes any overlapping Deletes

Filter

Black box that contains a match function that returns true/ false

Filters are to be used in cases where there is little or no opportunity for early-out optimizations.

As the last check, they can be stateful to implement limits.

Filter.match(cur)

`True` if cur should be included in result
`False` if cur should not be included in result