ADDB: MetaKeys

연세대학교 컴퓨터과학과 박상현

2019년 8월

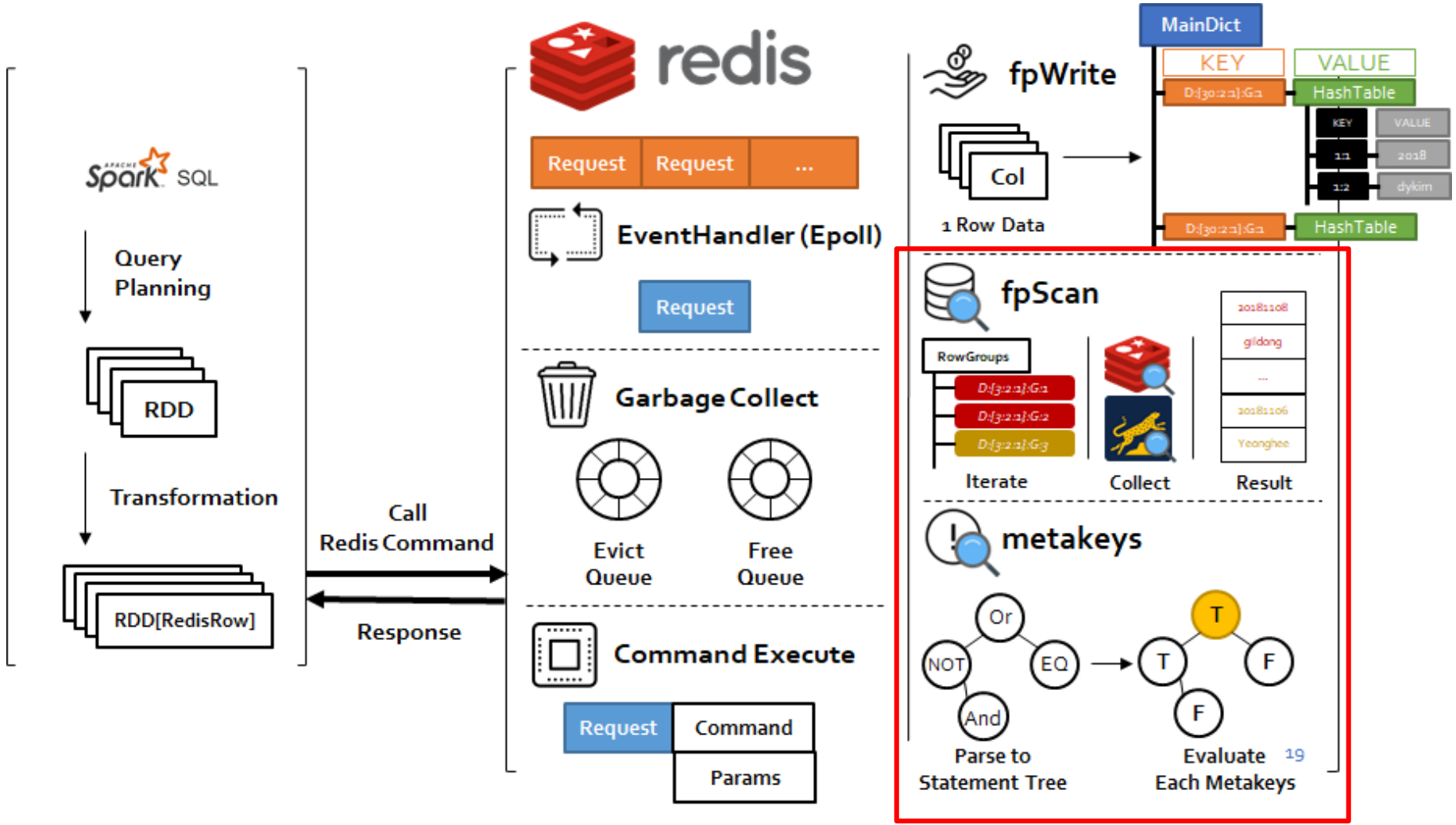
과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

과학기술정보통신부
Ministry of Science and ICT

연세대학교
YONSEI UNIVERSITY

정보통신기술진흥센터
Institute for Information & communications Technology Promotion


Spark SQL
Query Planning
RDD
Transformation
RDD[RedisRow]
Call Redis Command
Response
redis
Request
Request
...
EventHandler (Epoll)
Request
Garbage Collect
Evict Queue
Free Queue
Command Execute
Request
Command
Params
fpWrite
Col
1 Row Data
MainDict
KEY
VALUE
D:[30:21]:G:1
HashTable
KEY
VALUE
1:1
2018
1:2
dykim
D:[30:21]:G:1
HashTable
fpScan
RowGroups
D:[3:2:1]:G:1
D:[3:2:1]:G:2
D:[3:2:1]:G:3
Iterate
Collect
20181108
gildong
...
20181106
Yeonghee
Result
metakeys
Or
NOT
EQ
And
Parse to Statement Tree
T
T
F
F
Evaluate Each Metakeys
19

Contents

metakeys

- Redis의 MetaDict에 저장되어 있는 Key들 중에, 특정 조건에 맞는 Key들만 Filtering하는 연산
  - MetaDict의 Metakey는 다음과 같은 구조를 가지고 있
    M:{TableID:Partitions}
    Partitions = ColumnID:Value:ColumnID:Value...
  - ex) M:{100:2:0}
    → Col2 = 0

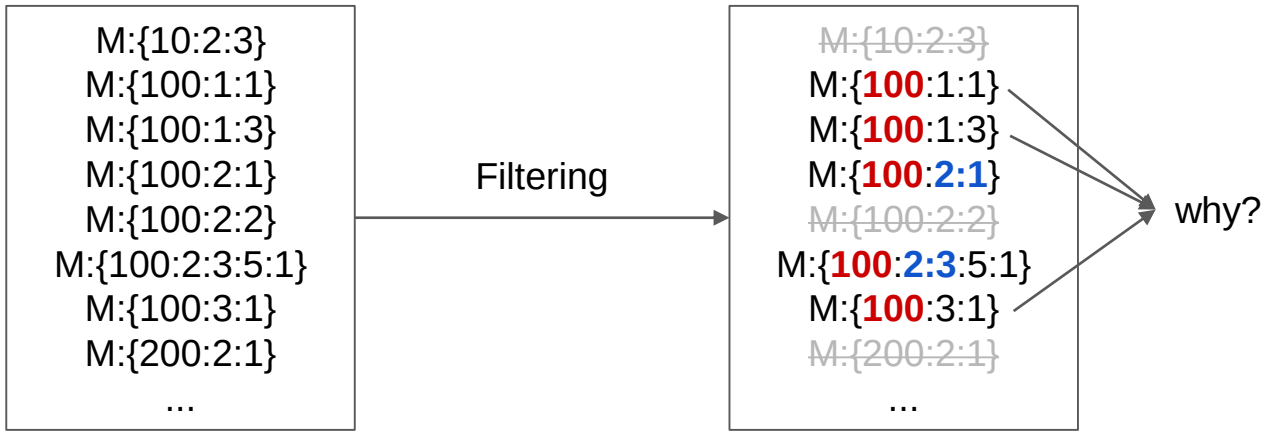
- Command Parameters
  - Pattern
    - ex) M:{100:*}
  - Statements
    - SQL의 Where절을 String으로 표현
    - ex) SELECT … WHERE **col2 = 3 OR col2 < 2**;
      → 3*2*EqualTo:2*2*LessThan:Or:$

metakeys

- Where Statements에 적합한 Partition을 가지고 있는 Metakey를 Filtering
  - ex) redis-cli> metakeys M:{100:*} 3*2*EqualTo:2*2*LessThan:Or:$
    - TableID

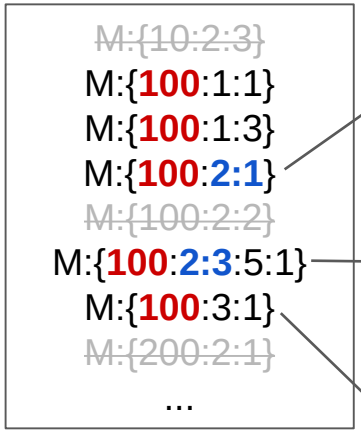
      100
    - Statements

      col2 = 3 OR col2 < 2



metakeys

- Q) 왜 M:{100:3:1}도 scan하는거에요?
  → 해당 Partition에, Statements 조건에 맞는 값들이 존재할 수 있음



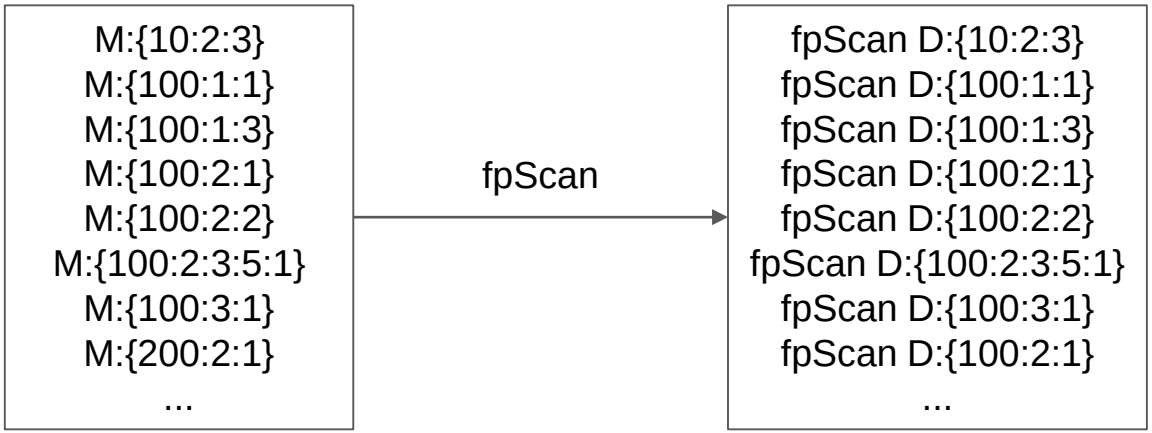
Col1	Col2	Col3	Col4	Col5
102	1	393	209	129
39	1	24	94	34
281	1	200	12	7

Col1	Col2	Col3	Col4	Col5
12	29	1	32	4
559	3	1	1332	123
321	1	1	1345	958

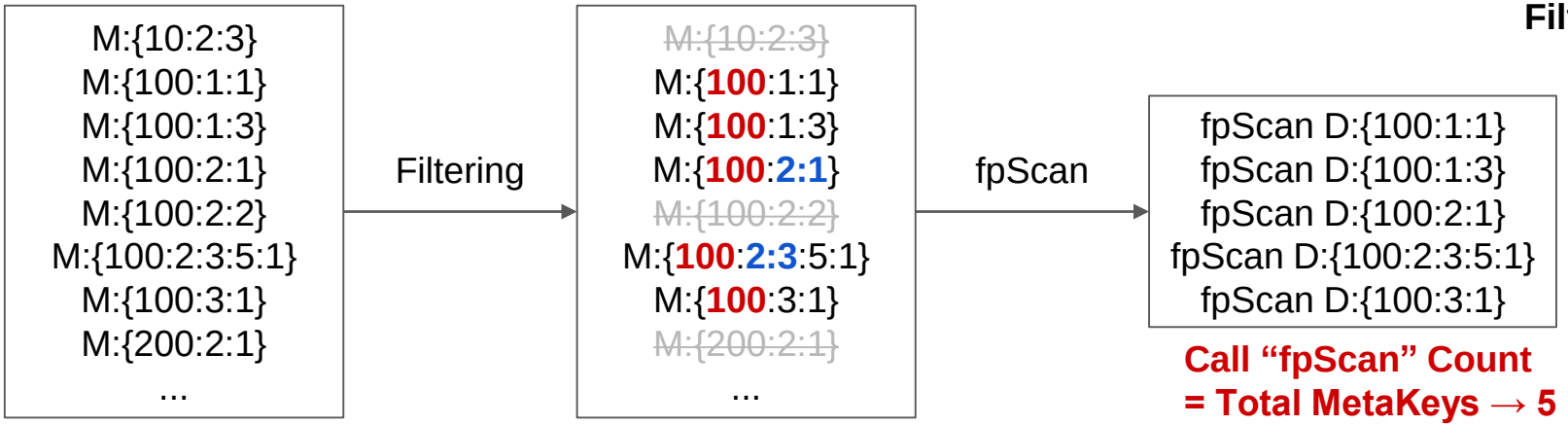
Col1	Col2	Col3	Col4	Col5
4	3	393	209	1
55	3	24	94	1
95	3	200	12	1

metakeys

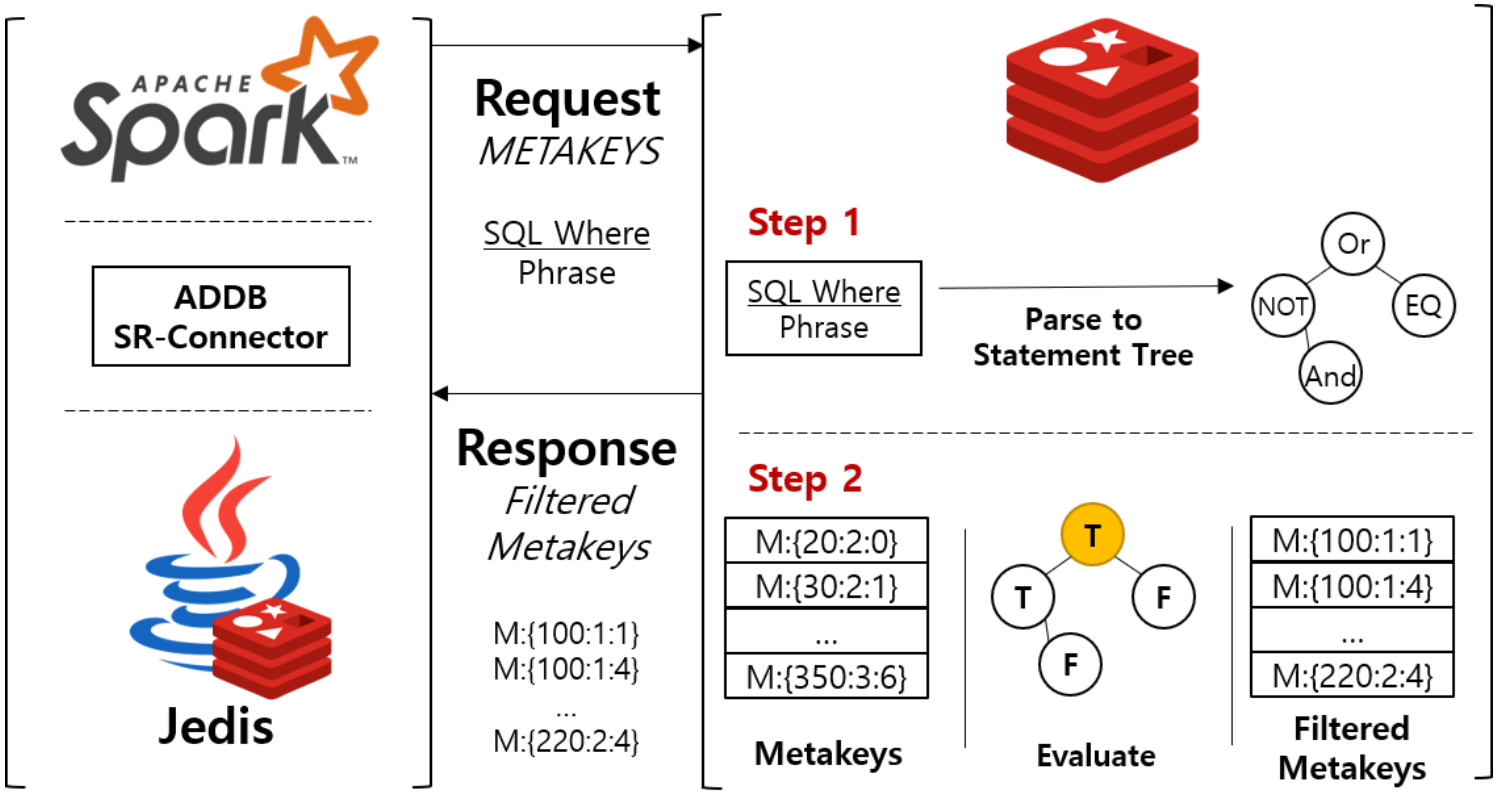
No Filtering



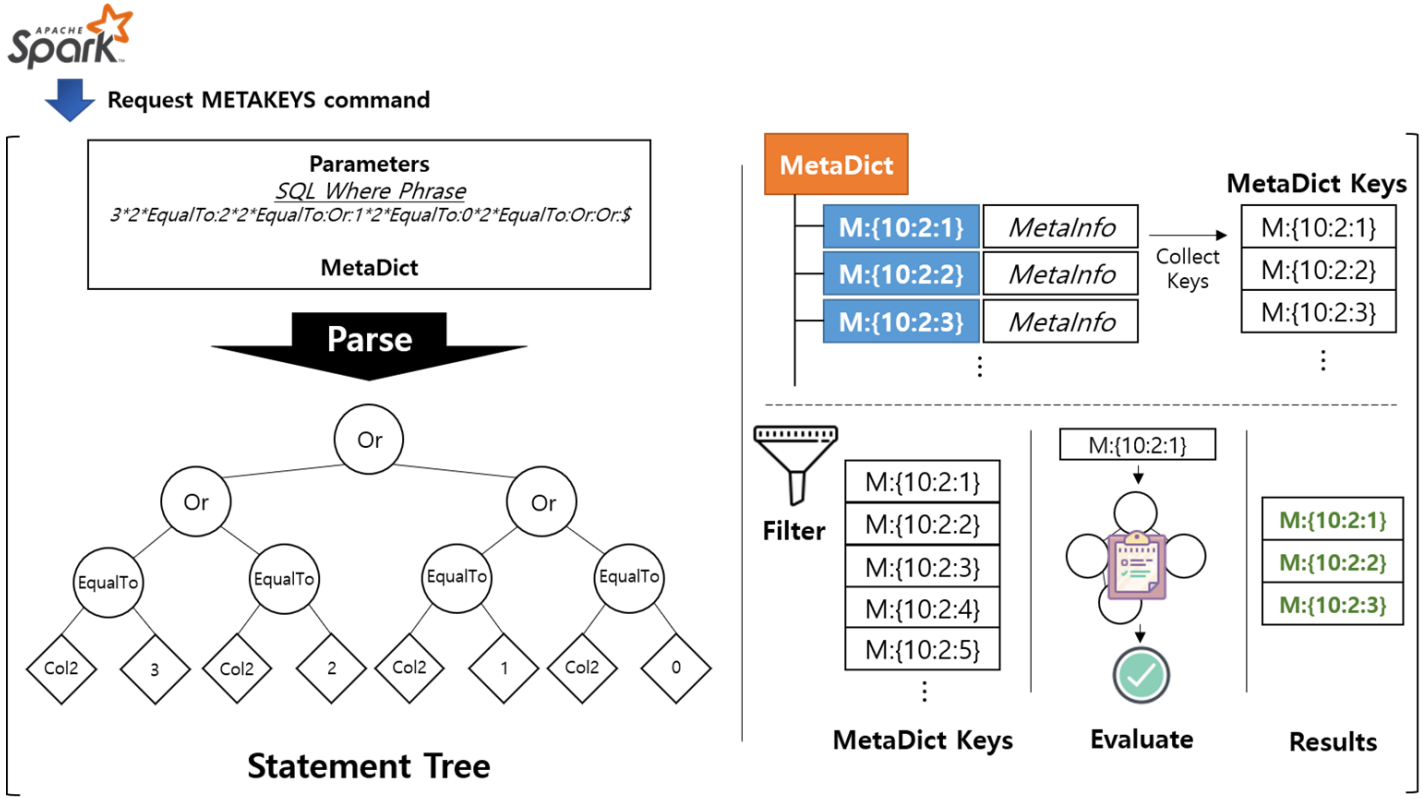
Filtering



metakeys - Overview



metakeys - Command Flow



metakeys - Code Structure

metakeysCommand()

Initialize Step

- MetaDict에서 MetaKey들을 Collect
- Pattern matching

Collect

Validate Step

- Stringfied Statements가 Valid Form인지 Check (미구현)

~~**Validate**~~

Parse & Evaluate Step

- Stringfied Statement를 Evaluation Tree로 Parse
- Evaluation Tree에 각 Metakey를 Evaluate, 조건에 맞으면 유지, 조건에 맞지 않으면 Filter

Parse
Evaluate
Filter

```
void metakeysCommand(client *c){
    /*Parses stringfied stack structure to readable parameters*/
    sds pattern = (sds) c->argv[1]->ptr;
    bool allkeys = (pattern[0] == '*' && pattern[1] == '\0');

    /*Initializes Vector for Metadict keys*/
    Vector metakeys;
    vectorTypeInit(&metakeys, STL_TYPE_SDS);

    /*Pattern match searching for metakeys*/
    ...

    /* If rawStatements is null or empty, prints pattern matching
     * results only...
     */
    ...

    sds rawStatementsStr = (sds) c->argv[2]->ptr;

    if (!validateStatements(rawStatementsStr)) {
        serverLog(LL_WARNING, "[FILTER] Stack structure is not valid form:
                  rawStatementsStr);
        addReplyErrorFormat(c, "[FILTER] Stack structure is not valid form:
                            rawStatementsStr);
        return;
    }

    /*Search each statmenet*/
    token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
    while (token != NULL) {
        // Parse
        ...

        // Evaluate
        ...

        // Filter
        ...
    }

    /*Prints out target partitions*/
    /*Scan data to client*/
    _addReplyMetakeysResults(c, &metakeys);
    vectorFree(&metakeys);
}
```

metakeys - Example

- ex) redis-cli> metakeys M:{100:*} 3*2*EqualTo:2*2*LessThan:Or:$

Evaluation Tree

MetaDict

M:{10:2:3}	hashdict
M:{100:1:1}	hashdict
M:{100:1:3:2:0}	hashdict
M:{100:2:1:3:7}	hashdict
M:{100:2:2}	hashdict
M:{100:2:3:5:1}	hashdict
M:{100:3:1}	hashdict
M:{100:3:2:4:1}	hashdict
M:{200:2:1}	hashdict
M:{200:2:3}	hashdict

Results

metakeys - Code Structure

metakeysCommand()

Initialize Step

- MetaDict에서 MetaKey들을 Collect
- Pattern matching

Collect

Validate Step

- Stringfied Statements가 Valid Form인지 Check (미구현)

~~**Validate**~~

Parse & Evaluate Step

- Stringfied Statement를 Evaluation Tree로 Parse
- Evaluation Tree에 각 Metakey를 Evaluate, 조건에 맞으면 유지, 조건에 맞지 않으면 Filter

Parse
Evaluate
Filter

```
void metakeysCommand(client *c){
    /*Parses stringfied stack structure to readable parameters*/
    sds pattern = (sds) c->argv[1]->ptr;
    bool allkeys = (pattern[0] == '*' && pattern[1] == '\0');

    /*Initializes Vector for Metadict keys*/
    Vector metakeys;
    vectorTypeInit(&metakeys, STL_TYPE_SDS);

    /*Pattern match searching for metakeys*/
    ...

    /* If rawStatements is null or empty, prints pattern matching
     * results only...
     */
    ...

    sds rawStatementsStr = (sds) c->argv[2]->ptr;

    if (!validateStatements(rawStatementsStr)) {
        serverLog(LL_WARNING, "[FILTER] Stack structure is not valid form:
                  rawStatementsStr);
        addReplyErrorFormat(c, "[FILTER] Stack structure is not valid form:
                            rawStatementsStr);
        return;
    }

    /*Search each statmenet*/
    token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
    while (token != NULL) {
        // Parse
        ...

        // Evaluate
        ...

        // Filter
        ...
    }

    /*Prints out target partitions*/
    /*Scan data to client*/
    _addReplyMetakeysResults(c, &metakeys);
    vectorFree(&metakeys);
}
```

metakeys - Initialize Step

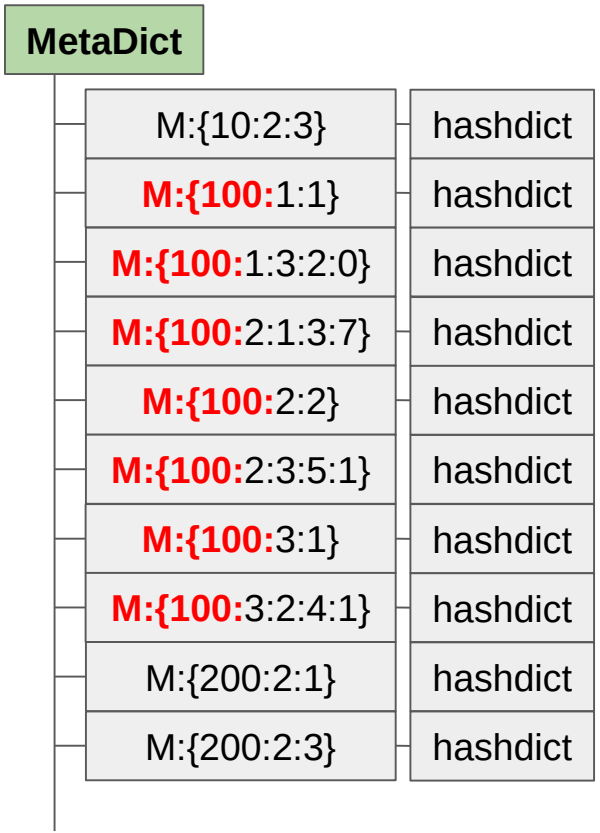
- MetaDict에 있는 모든 metakey들을 Vector로 수
  집함
- 이때, Pattern matching을 수행하여, Pattern에 맞
  는 metakey들만 가져옴
- Stringfied Statements가 주어지지 않으면,
  Pattern matching한 결과만 Reply

```
/*Parses stringfied stack structure to readable parameters*/
sds pattern = (sds) c->argv[1]->ptr;
bool allkeys = (pattern[0] == '*' && pattern[1] == '\0');

/*Initializes Vector for Metadict keys*/
Vector metakeys;
vectorTypeInit(&metakeys, STL_TYPE_SDS);

/*Pattern match searching for metakeys*/
dictIterator *di = dictGetSafeIterator(c->db->Metadict);
dictEntry *de = NULL;
while ((de = dictNext(di)) != NULL) {
    sds metakey = (sds) dictGetKey(de);
    if (
        allkeys ||
        stringmatchlen(pattern, sdslen(pattern), metakey,
                       sdslen(metakey), 0)
    ) {
        vectorAdd(&metakeys, metakey);
    }
}
dictReleaseIterator(di);

/* If rawStatements is null or empty, prints pattern matching
 * results only...
 */
if (c->argc < 3) {
    /*Prints out target partitions*/
    /*Scan data to client*/
    _addReplyMetakeysResults(c, &metakeys);
    vectorFree(&metakeys);
    return;
}
```

metakeys - Example

- ex) redis-cli> metakeys **M:{100:*}** 3*2*EqualTo:2*2*LessThan:Or:$

Evaluation Tree



Results

M:{100:1:1}
M:{100:1:3:2:0}
M:{100:2:1:3:7}
M:{100:2:2}

M:{100:2:3:5:1}
M:{100:3:1}
M:{100:3:2:4:1}

metakeys - Code Structure

metakeysCommand()

Initialize Step

- MetaDict에서 MetaKey들을 Collect
- Pattern matching

Validate Step

- Stringfied Statements가 Valid Form인지 Check (미구현)

Parse & Evaluate Step

- Stringfied Statement를 Evaluation Tree로 Parse
- Evaluation Tree에 각 Metakey를 Evaluate, 조건에 맞으면 유지, 조건에 맞지 않으면 Filter

Collect

Validate

Parse
Evaluate
Filter

```
void metakeysCommand(client *c){
    /*Parses stringfied stack structure to readable parameters*/

// TODO(totoro): Implements validateStatements function by regex.
bool validateStatements(const sds rawStatementsStr) {
    return true;
}

// TODO(totoro): Implements validateStatement function by regex.
bool validateStatement(const sds rawStatementStr) {
    return true;
}
    */
    ...

    sds rawStatementsStr = (sds) c->argv[2]->ptr;

    if (!validateStatements(rawStatementsStr)) {
        serverLog(LL_WARNING, "[FILTER] Stack structure is not valid form:
                  rawStatementsStr);
        addReplyErrorFormat(c, "[FILTER] Stack structure is not valid form:
                            rawStatementsStr);
        return;
    }

    /*Search each statmenet*/
    token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
    while (token != NULL) {
        // Parse
        ...

        // Evaluate
        ...

        // Filter
        ...
    }

    /*Prints out target partitions*/
    /*Scan data to client*/
    _addReplyMetakeysResults(c, &metakeys);
    vectorFree(&metakeys);
}
```

metakeys - Code Structure

metakeysCommand()

Initialize Step

- MetaDict에서 MetaKey들을 Collect
- Pattern matching

Collect

Validate Step

- Stringfied Statements가 Valid Form인지 Check (미구현)

~~**Validate**~~

Parse & Evaluate Step

- Stringfied Statement를 Evaluation Tree로 Parse
- Evaluation Tree에 각 Metakey를 Evaluate, 조건에 맞으면 유지, 조건에 맞지 않으면 Filter

Parse
Evaluate
Filter

```
void metakeysCommand(client *c){
    /*Parses stringfied stack structure to readable parameters*/
    sds pattern = (sds) c->argv[1]->ptr;
    bool allkeys = (pattern[0] == '*' && pattern[1] == '\0');

    /*Initializes Vector for Metadict keys*/
    Vector metakeys;
    vectorTypeInit(&metakeys, STL_TYPE_SDS);

    /*Pattern match searching for metakeys*/
    ...

    /* If rawStatements is null or empty, prints pattern matching
     * results only...
     */
    ...

    sds rawStatementsStr = (sds) c->argv[2]->ptr;

    if (!validateStatements(rawStatementsStr)) {
        serverLog(LL_WARNING, "[FILTER] Stack structure is not valid form:
                  rawStatementsStr);
        addReplyErrorFormat(c, "[FILTER] Stack structure is not valid form:
                            rawStatementsStr);
        return;
    }

    /*Search each statmenet*/
    token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
    while (token != NULL) {
        // Parse
        ...

        // Evaluate
        ...

        // Filter
        ...
    }

    /*Prints out target partitions*/
    /*Scan data to client*/
    _addReplyMetakeysResults(c, &metakeys);
    vectorFree(&metakeys);
}
```

metakeys - Parse & Evaluate Step

- Related Structs
  - Condition
    - Where의 하나의 Condition을 저장하는 구조
    - ex) 3*2*EqualTo:
  - ConditionChild
    - Condition의 Child에 해당하는 Value를 저장함
    - Long type, Sds type, Condition type
    - ex) 3 (long)
  - ConditionValue (union)
    - Condition 혹은 Long 혹은 sds
  - PartitionParameter
    - Metakey의 Partition 정보를 저장하는 구조
    - ex) M:{100:2:3}

      →

      columnId = 2, PartitionValue = 3

```
/*Partition Filter Parameters*/
typedef union _ConditionValue {
    void *cond;
    long l;
    sds s;
} ConditionValue;

typedef struct _ConditionChild {
    unsigned type:2;
    ConditionValue value;
} ConditionChild;

typedef struct _Condition {
    unsigned op:4;  // Operator
    int opCount;
    bool isLeaf;
    ConditionChild *first;
    ConditionChild *second;
} Condition;

typedef struct _PartitionValue {
    long l;
    sds s;
} PartitionValue;

typedef struct _PartitionParameter {
    int columnId;
    unsigned type:2;
    PartitionValue value;
} PartitionParameter;
```

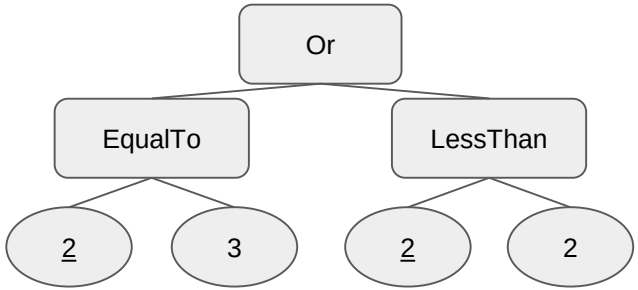
metakeys - Parse & Evaluate Step

- Stringfied Statements를 하나씩 Parse & Evaluate
  - Statement Suffix(“**$**”)로 tokenize
  - ex)
    3*2*EqualTo:2*2*LessThan:Or:**$**4*5*EqualTo:**$**
    →
    3*2*EqualTo:2*2*LessThan:Or:
    4*5*EqualTo:
  - 하나의 Statement에 대해서 Parse & Evaluate함

```
char copyStr[sdslen(rawStatementsStr) + 1];
char *savePtr = NULL;
char *token = NULL;
memcpy(copyStr, rawStatementsStr, sdslen(rawStatementsStr) + 1);

token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
while (token != NULL) {
    Condition *root;
    sds rawStatementStr = sdsnew(token);

    if (parseStatement(rawStatementStr, &root) == C_ERR) {
        serverLog(
                LL_WARNING,
                "[FILTER][FATAL] Stack condition parser failed, server wo
                rawStatementsStr);
        addReplyErrorFormat(
                c,
                "[FILTER][FATAL] Stack condition parser failed, server wo
                rawStatementsStr);
        return;
    }

    // serverLog(LL_DEBUG, "   ");
    // serverLog(LL_DEBUG, "[FILTER][PARSE] Condition Tree");
    // logCondition(root);

    Vector filteredMetakeys;
    vectorInit(&filteredMetakeys);
    for (size_t i = 0; i < vectorCount(&metakeys); ++i) {
        sds metakey = (sds) vectorGet(&metakeys, i);
        if (evaluateCondition(root, metakey)) {
            vectorAdd(&filteredMetakeys, metakey);
        }
    }
    vectorFree(&metakeys);
    metakeys = filteredMetakeys;

    sdsfree(rawStatementStr);
    freeConditions(root);

    token = strtok_r(NULL, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
}
```

metakeys - Parse & Evaluate Step

- Stringfied Statement (rawStatementStr)
  - 하나의 Statement를 Statement Tree로 Parse함
  - parseStatement() function에서 수행함
  - Condition *root에 Statement Tree의 root를 저장
  - ex)
    3*2*EqualTo:2*2*LessThan:Or:
    →




```
char copyStr[sdslen(rawStatementsStr) + 1];
char *savePtr = NULL;
char *token = NULL;
memcpy(copyStr, rawStatementsStr, sdslen(rawStatementsStr) + 1);

token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
while (token != NULL) {
    Condition *root;
    sds rawStatementStr = sdsnew(token);

    if (parseStatement(rawStatementStr, &root) == C_ERR) {
        serverLog(
                LL_WARNING,
                "[FILTER][FATAL] Stack condition parser failed, server wo
                rawStatementsStr);
        addReplyErrorFormat(
                c,
                "[FILTER][FATAL] Stack condition parser failed, server wo
                rawStatementsStr);
        return;
    }

    // serverLog(LL_DEBUG, "   ");
    // serverLog(LL_DEBUG, "[FILTER][PARSE] Condition Tree");
    // logCondition(root);

    Vector filteredMetakeys;
    vectorInit(&filteredMetakeys);
    for (size_t i = 0; i < vectorCount(&metakeys); ++i) {
        sds metakey = (sds) vectorGet(&metakeys, i);
        if (evaluateCondition(root, metakey)) {
            vectorAdd(&filteredMetakeys, metakey);
        }
    }
    vectorFree(&metakeys);
    metakeys = filteredMetakeys;

    sdsfree(rawStatementStr);
    freeConditions(root);

    token = strtok_r(NULL, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
}
```

metakeys - Parse Statement

- Stack을 사용하여 rawStatement를 Tree로 Parse
  - Why?
    - Tree로 만들어서 사용하게 되면, metakey를 하나씩 evaluate할 때 Condition을 매번 재사용할 수 있기 때문
    - Stack으로 evaluate하게 되면, metakey를 하나씩 evaluate할 때마다 stack 연산이 필요함 (memory write 발생)

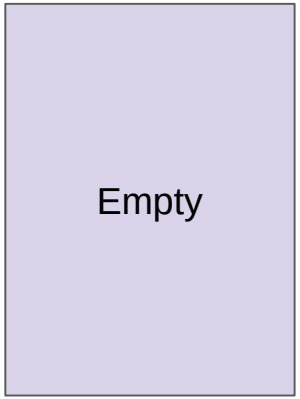
- ex) 3*2*EqualTo:2*2*LessThan:Or:

```
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

- Stack을 초기화하고, Operator delimiter(“**:**”)로 tokenize함

Token	savePtr (Remain)
3*2*EqualTo	2*2*LessThan:Or:



Stack

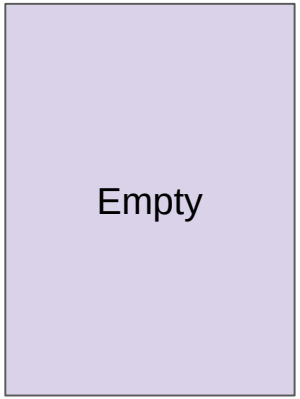
```c
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

- Stack 상황과 Token에 따라서, Condition을 생성함
  - createCondition() function으로 condition 생성

Token (= rawCondition)

3*2*EqualTo	2*2*LessThan:Or:

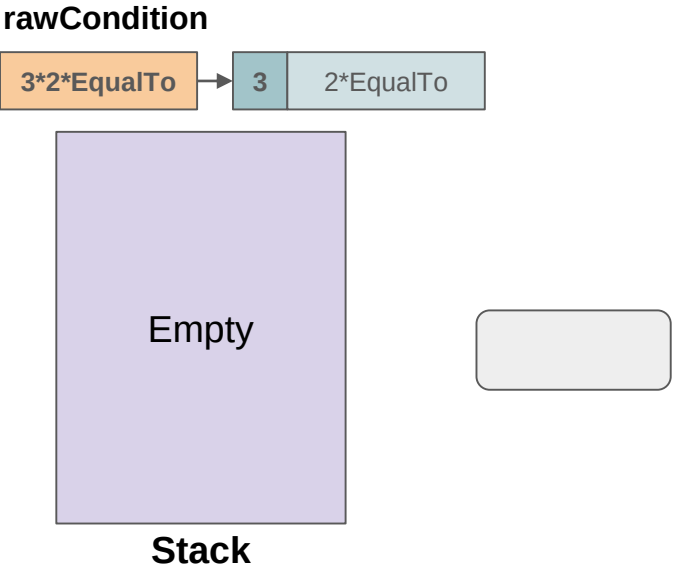


Stack

```c
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

- rawCondition을 Operand Delimiter(“*”)로 Tokenize




```c
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);
int createCondition(const char *rawConditionStr, Stack *s,
                    Condition **cond) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawConditionStr, strlen(rawConditionStr) + 1);
    serverLog(LL_DEBUG, "[FILTER][PARSE] raw condition: %s", rawConditionStr);

    // Parses raw condition by operand unit.
    Condition *newcond = zmalloc(sizeof(Condition));
    token = strtok_r(copyStr, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    int leafOperandCount = 0;
    while (savePtr[0] != '\0') {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        long tokenLong;
        if (string2l(token, strlen(token), &tokenLong) == 1) {
            child->type = CONDITION_CHILD_VALUE_TYPE_LONG;
            child->value.l = tokenLong;
        } else {
            child->type = CONDITION_CHILD_VALUE_TYPE_SDS;
            child->value.s = sdsnew(token);
        }
        stackPush(s, (void *) child);
        leafOperandCount++;
        token = strtok_r(NULL, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    }
    serverLog(LL_DEBUG, "[FILTER][PARSE] condition operator: %s", token);
    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

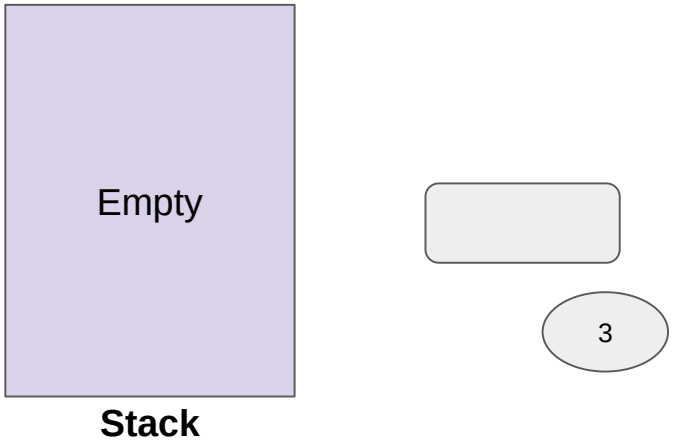
- Operand(ConditionChild) 객체 생성
  - Long type이면, type을 Long으로 지정한 뒤 value.l로 저장
  - sds type이면, type을 sds로 지정한 뒤, value.s로 저장

```
int createCondition(const char *rawConditionStr, Stack *s,
                    Condition **cond) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawConditionStr, strlen(rawConditionStr) + 1);
    serverLog(LL_DEBUG, "[FILTER][PARSE] raw condition: %s", rawConditionStr);

    // Parses raw condition by operand unit.
    Condition *newcond = zmalloc(sizeof(Condition));
    token = strtok_r(copyStr, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    int leafOperandCount = 0;
    while (savePtr[0] != '\0') {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        long tokenLong;
        if (string2l(token, strlen(token), &tokenLong) == 1) {
            child->type = CONDITION_CHILD_VALUE_TYPE_LONG;
            child->value.l = tokenLong;
        } else {
            child->type = CONDITION_CHILD_VALUE_TYPE_SDS;
            child->value.s = sdsnew(token);
        }
        stackPush(s, (void *) child);
        leafOperandCount++;
        token = strtok_r(NULL, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    }
    serverLog(LL_DEBUG, "[FILTER][PARSE] condition operator: %s", token);
```

rawCondition

3	2*EqualTo

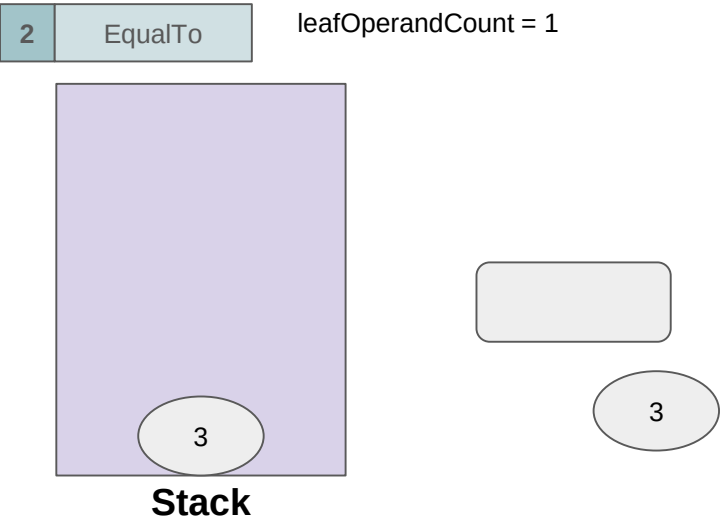
leafOperandCount = 0



metakeys - Parse Statement

- Stack에 ConditionChild를 Push
  - leafOperandCount를 1 증가시킴
  - Operand Delimiter로 다시 Tokenize

rawCondition




```
int createCondition(const char *rawConditionStr, Stack *s,
                    Condition **cond) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawConditionStr, strlen(rawConditionStr) + 1);
    serverLog(LL_DEBUG, "[FILTER][PARSE] raw condition: %s", rawConditionStr);

    // Parses raw condition by operand unit.
    Condition *newcond = zmalloc(sizeof(Condition));
    token = strtok_r(copyStr, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    int leafOperandCount = 0;
    while (savePtr[0] != '\0') {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        long tokenLong;
        if (string2l(token, strlen(token), &tokenLong) == 1) {
            child->type = CONDITION_CHILD_VALUE_TYPE_LONG;
            child->value.l = tokenLong;
        } else {
            child->type = CONDITION_CHILD_VALUE_TYPE_SDS;
            child->value.s = sdsnew(token);
        }
        stackPush(s, (void *) child);
        leafOperandCount++;
        token = strtok_r(NULL, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    }
    serverLog(LL_DEBUG, "[FILTER][PARSE] condition operator: %s", token);
```

metakeys - Parse Statement

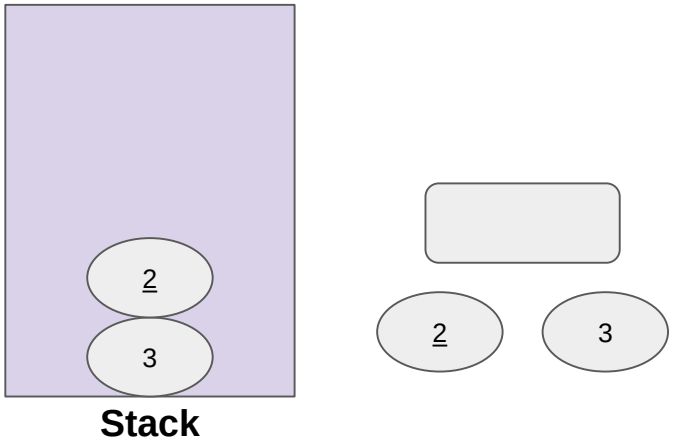
- Keep Going… Finish

```
int createCondition(const char *rawConditionStr, Stack *s,
                    Condition **cond) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawConditionStr, strlen(rawConditionStr) + 1);
    serverLog(LL_DEBUG, "[FILTER][PARSE] raw condition: %s", rawConditionStr);

    // Parses raw condition by operand unit.
    Condition *newcond = zmalloc(sizeof(Condition));
    token = strtok_r(copyStr, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    int leafOperandCount = 0;
    while (savePtr[0] != '\0') {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        long tokenLong;
        if (string2l(token, strlen(token), &tokenLong) == 1) {
            child->type = CONDITION_CHILD_VALUE_TYPE_LONG;
            child->value.l = tokenLong;
        } else {
            child->type = CONDITION_CHILD_VALUE_TYPE_SDS;
            child->value.s = sdsnew(token);
        }
        stackPush(s, (void *) child);
        leafOperandCount++;
        token = strtok_r(NULL, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    }
    serverLog(LL_DEBUG, "[FILTER][PARSE] condition operator: %s", token);
```

rawCondition

EqualTo	\0

leafOperandCount = 2



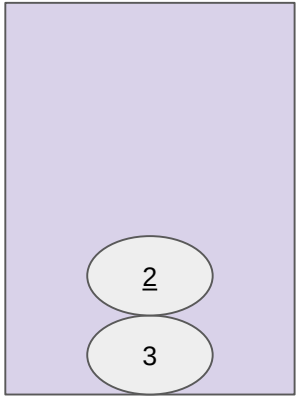
metakeys - Parse Statement

- Continue on createCondition()
  - 남은 Token의 OperatorType Check
  - optype에 따른 child 개수인 opCount Check
    - EqualTo는 opCount = 2

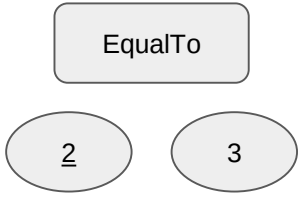
rawCondition

EqualTo	\0

leafOperandCount = 2



Stack




```
int optype = _getOperatorType(token);
if (optype == -1) {
    serverLog(LL_WARNING, "[FILTER][PARSE][FATAL] Invalid Operator '%s'",
              token);
    serverPanic("[FILTER][PARSE][FATAL] Invalid Operator");
    return C_ERR;
}
newcond->op = optype;
newcond->opCount = _getOperatorOperandCount(optype);

if (leafOperandCount > 0) {
    if (leafOperandCount != newcond->opCount) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = true;
    newcond->first = (ConditionChild *) stackPop(s);
    if (newcond->first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
        // Invalid condition format detected...
        return C_ERR;
    }
    newcond->second = NULL;
    if (newcond->opCount == 2) {
        newcond->second = (ConditionChild *) stackPop(s);
    }
} else {
    if (newcond->opCount > (int) stackCount(s)) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = false;

    ConditionChild *child = (ConditionChild *) zmalloc(
            sizeof(ConditionChild));
    child->type = CONDITION_CHILD_VALUE_TYPE_COND;
    child->value.cond = (Condition *) stackPop(s);
    newcond->first = child;

    newcond->second = NULL;
    if (newcond->opCount == 2) {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        child->type = CONDITION_CHILD_VALUE_TYPE_COND;
        child->value.cond = (Condition *) stackPop(s);
        newcond->second = child;
    }
}
*cond = newcond;

return C_OK;
```

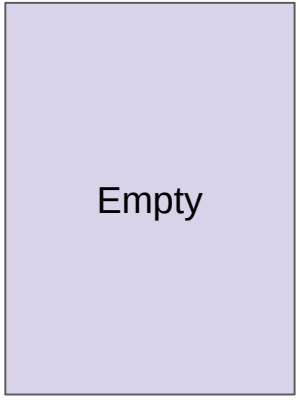
metakeys - Parse Statement

- Continue on createCondition()
  - leafOperandCount가 2 이므로, Leaf Condition
  - Stack에서 Pop하여 newcond에 붙여줌
  - 첫번째 element는 항상 columnID이므로 Err check

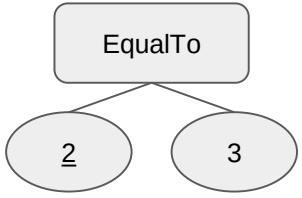
rawCondition

EqualTo	\0

leafOperandCount = 2



Stack



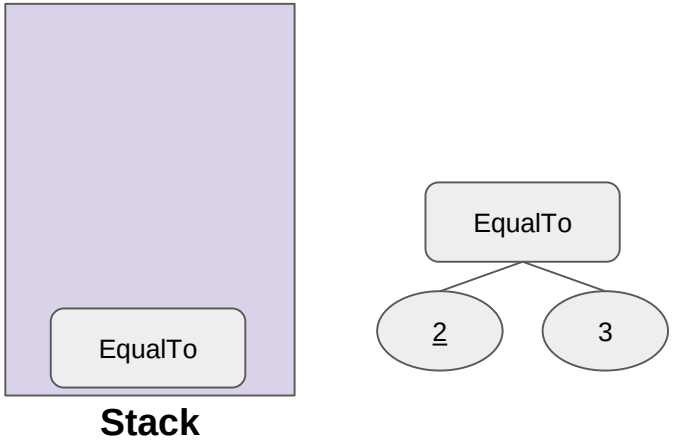

```
int optype = _getOperatorType(token);
if (optype == -1) {
    serverLog(LL_WARNING, "[FILTER][PARSE][FATAL] Invalid Operator '%s'",
              token);
    serverPanic("[FILTER][PARSE][FATAL] Invalid Operator");
    return C_ERR;
}
newcond->op = optype;
newcond->opCount = _getOperatorOperandCount(optype);

if (leafOperandCount > 0) {
    if (leafOperandCount != newcond->opCount) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = true;
    newcond->first = (ConditionChild *) stackPop(s);
    if (newcond->first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
        // Invalid condition format detected...
        return C_ERR;
    }
    newcond->second = NULL;
    if (newcond->opCount == 2) {
        newcond->second = (ConditionChild *) stackPop(s);
    }
} else {
    if (newcond->opCount > (int) stackCount(s)) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = false;

    ConditionChild *child = (ConditionChild *) zmalloc(
            sizeof(ConditionChild));
    child->type = CONDITION_CHILD_VALUE_TYPE_COND;
    child->value.cond = (Condition *) stackPop(s);
    newcond->first = child;

    newcond->second = NULL;
    if (newcond->opCount == 2) {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        child->type = CONDITION_CHILD_VALUE_TYPE_COND;
        child->value.cond = (Condition *) stackPop(s);
        newcond->second = child;
    }
}
*cond = newcond;

return C_OK;
```

metakeys - Parse Statement

- 새로 만든 Condition을 Stack에 넣고, 다음 Condition Parse...

Token (= rawCondition)

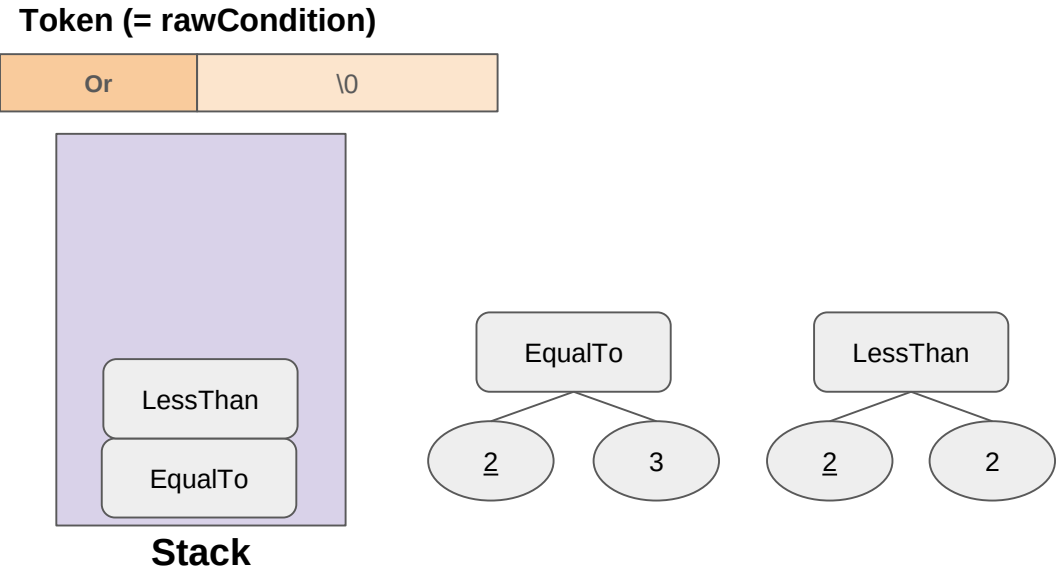
2*2*LessThan	Or:




```c
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

- 마지막 “Or”의 Case
  - rawCondition에 child가 없으므로 동작이 약간 다름

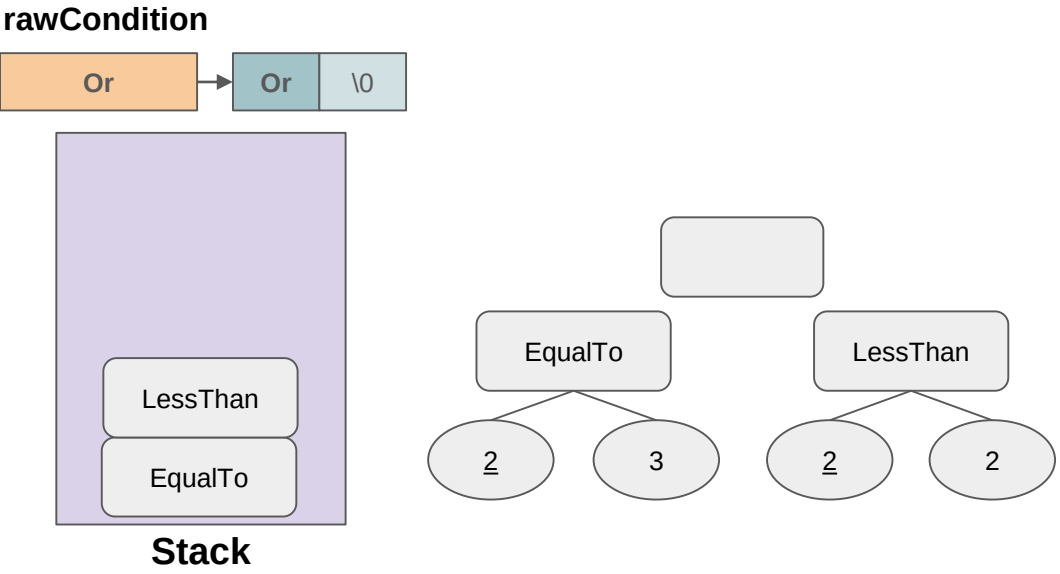



```
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

- rawCondition을 Operand Delimiter(“*”)로 Tokenize
  - savePtr이 이미 “\0”이므로 While문을 거치지 않음

```
int createCondition(const char *rawConditionStr, Stack *s,
                    Condition **cond) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawConditionStr, strlen(rawConditionStr) + 1);
    serverLog(LL_DEBUG, "[FILTER][PARSE] raw condition: %s", rawConditionStr);

    // Parses raw condition by operand unit.
    Condition *newcond = zmalloc(sizeof(Condition));
    token = strtok_r(copyStr, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    int leafOperandCount = 0;
    while (savePtr[0] != '\0') {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        long tokenLong;
        if (string2l(token, strlen(token), &tokenLong) == 1) {
            child->type = CONDITION_CHILD_VALUE_TYPE_LONG;
            child->value.l = tokenLong;
        } else {
            child->type = CONDITION_CHILD_VALUE_TYPE_SDS;
            child->value.s = sdsnew(token);
        }
        stackPush(s, (void *) child);
        leafOperandCount++;
        token = strtok_r(NULL, PARTITION_FILTER_OPERAND_DELIMITER, &savePtr);
    }
    serverLog(LL_DEBUG, "[FILTER][PARSE] condition operator: %s", token);
```



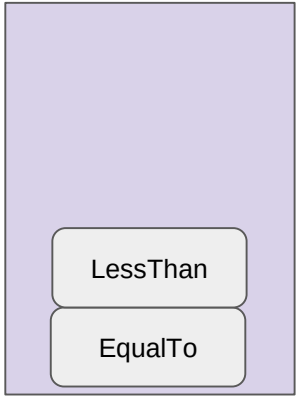
metakeys - Parse Statement

- Continue on createCondition()
  - 남은 Token의 OperatorType Check
  - optype에 따른 child 개수인 opCount Check
    - Or는 opCount = 2

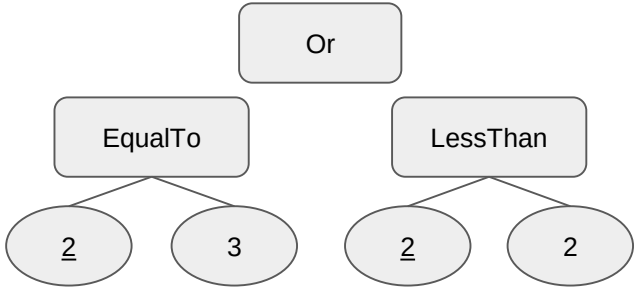
rawCondition

Or	\0

leafOperandCount = 0



Stack




```
int optype = _getOperatorType(token);
if (optype == -1) {
    serverLog(LL_WARNING, "[FILTER][PARSE][FATAL] Invalid Operator '%s'",
              token);
    serverPanic("[FILTER][PARSE][FATAL] Invalid Operator");
    return C_ERR;
}
newcond->op = optype;
newcond->opCount = _getOperatorOperandCount(optype);

if (leafOperandCount > 0) {
    if (leafOperandCount != newcond->opCount) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = true;
    newcond->first = (ConditionChild *) stackPop(s);
    if (newcond->first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
        // Invalid condition format detected...
        return C_ERR;
    }
    newcond->second = NULL;
    if (newcond->opCount == 2) {
        newcond->second = (ConditionChild *) stackPop(s);
    }
} else {
    if (newcond->opCount > (int) stackCount(s)) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = false;

    ConditionChild *child = (ConditionChild *) zmalloc(
            sizeof(ConditionChild));
    child->type = CONDITION_CHILD_VALUE_TYPE_COND;
    child->value.cond = (Condition *) stackPop(s);
    newcond->first = child;

    newcond->second = NULL;
    if (newcond->opCount == 2) {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        child->type = CONDITION_CHILD_VALUE_TYPE_COND;
        child->value.cond = (Condition *) stackPop(s);
        newcond->second = child;
    }
}
*cond = newcond;

return C_OK;
```

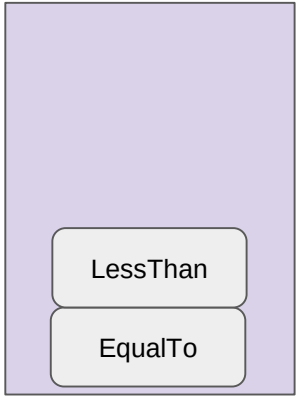
metakeys - Parse Statement

- Continue on createCondition()
  - leafOperandCount가 0 이므로, Non-Leaf Condition
  - Condition들을 Stack에서 Pop하여 Child로 붙여줌

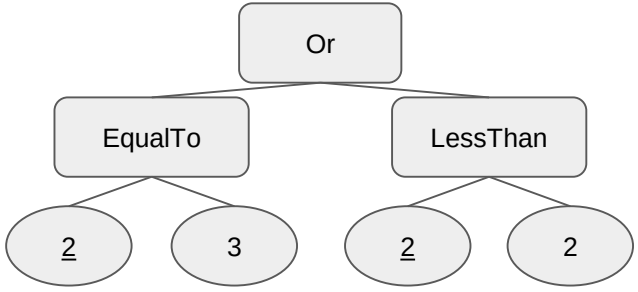
rawCondition

Or	\0

leafOperandCount = 0



Stack

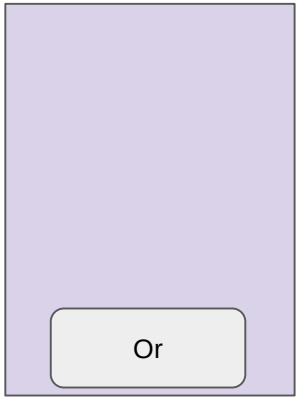



```
int optype = _getOperatorType(token);
if (optype == -1) {
    serverLog(LL_WARNING, "[FILTER][PARSE][FATAL] Invalid Operator '%s'",
              token);
    serverPanic("[FILTER][PARSE][FATAL] Invalid Operator");
    return C_ERR;
}
newcond->op = optype;
newcond->opCount = _getOperatorOperandCount(optype);

if (leafOperandCount > 0) {
    if (leafOperandCount != newcond->opCount) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = true;
    newcond->first = (ConditionChild *) stackPop(s);
    if (newcond->first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
        // Invalid condition format detected...
        return C_ERR;
    }
    newcond->second = NULL;
    if (newcond->opCount == 2) {
        newcond->second = (ConditionChild *) stackPop(s);
    }
} else {
    if (newcond->opCount > (int) stackCount(s)) {
        // Invalid condition format detected...
        return C_ERR;
    }

    newcond->isLeaf = false;

    ConditionChild *child = (ConditionChild *) zmalloc(
            sizeof(ConditionChild));
    child->type = CONDITION_CHILD_VALUE_TYPE_COND;
    child->value.cond = (Condition *) stackPop(s);
    newcond->first = child;

    newcond->second = NULL;
    if (newcond->opCount == 2) {
        ConditionChild *child = (ConditionChild *) zmalloc(
                sizeof(ConditionChild));
        child->type = CONDITION_CHILD_VALUE_TYPE_COND;
        child->value.cond = (Condition *) stackPop(s);
        newcond->second = child;
    }
}
*cond = newcond;

return C_OK;
```

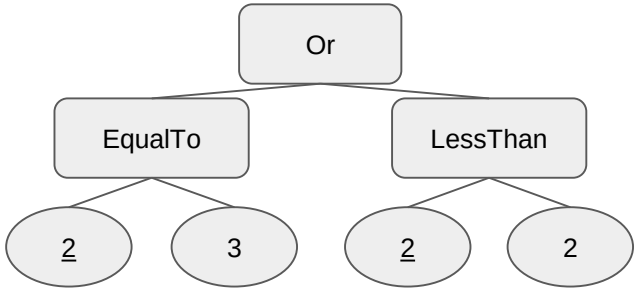
metakeys - Parse Statement

- 새로 만든 Condition을 Stack에 넣고, 다음 Condition Parse…
  - rawStatementStr을 모두 Parse했으므로 마지막 root만 Stack에 남음

Token (= rawCondition)



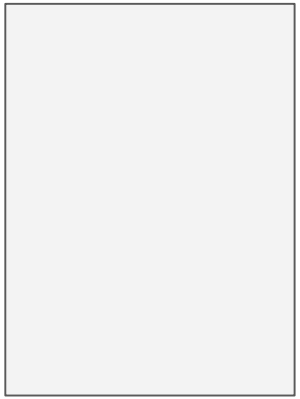
Stack



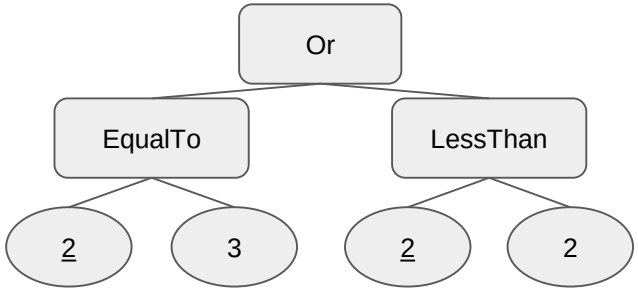

```
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

metakeys - Parse Statement

- Root를 Stack에서 Pop하고 stack을 free시킨 뒤, Terminate...



~~Stack~~




```c
int parseStatement(const sds rawStatementStr, Condition **root) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    memcpy(copyStr, rawStatementStr, sdslen(rawStatementStr) + 1);

    Stack s;
    stackInit(&s);

    // Parses raw conditions by condition unit.
    token = strtok_r(copyStr, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    while (token != NULL) {
        Condition *cond;
        if (createCondition(token, &s, &cond) == C_ERR) {
            serverLog(LL_DEBUG,
                      "[FILTER][PARSE] Input Condition is invalid form: %s",
                      token);
            return C_ERR;
        }
        stackPush(&s, (void *) cond);
        token = strtok_r(NULL, PARTITION_FILTER_OPERATOR_DELIMITER, &savePtr);
    }

    // Last token must be NULL.
    // Number of stack element must be 1.
    if (token != NULL || stackCount(&s) != 1) {
        serverLog(LL_DEBUG,
                  "[FILTER][PARSE] Entire condition is invalid form: lastToken
                  token, stackCount(&s));
        return C_ERR;
    }

    *root = stackPop(&s);
    stackFree(&s);

    return C_OK;
}
```

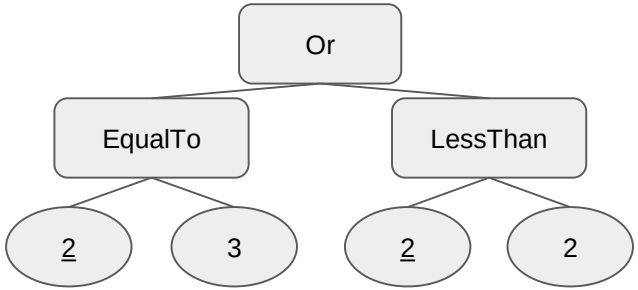
metakeys - Example

- ex) redis-cli> metakeys M:{100:*} 3*2*EqualTo:2*2*LessThan:Or:$

MetaDict

M:{10:2:3}	hashdict
M:{100:1:1}	hashdict
M:{100:1:3:2:0}	hashdict
M:{100:2:1:3:7}	hashdict
M:{100:2:2}	hashdict
M:{100:2:3:5:1}	hashdict
M:{100:3:1}	hashdict
M:{100:3:2:4:1}	hashdict
M:{200:2:1}	hashdict
M:{200:2:3}	hashdict

Evaluation Tree



Results

M:{100:1:1}
M:{100:1:3:2:0}
M:{100:2:1:3:7}
M:{100:2:2}

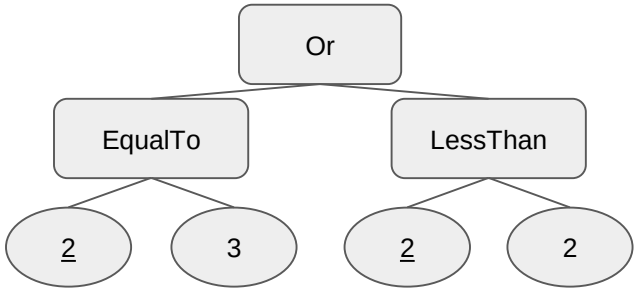
M:{100:2:3:5:1}
M:{100:3:1}
M:{100:3:2:4:1}

metakeys - Parse & Evaluate Step

- Finished Parse Step… Next?
  - Collect한 MetaKey들을 Statement Tree에서 Evaluate
  - Metakey들을 하나씩 순회하며 Evaluate함
    - 통과된 Metakey만 결과에 저장함

M:{100:1:1}
M:{100:1:3:2:0}
M:{100:2:1:3:7}
M:{100:2:2}
M:{100:2:3:5:1}
M:{100:3:1}
M:{100:3:2:4:1}

Evaluate




```
char copyStr[sdslen(rawStatementsStr) + 1];
char *savePtr = NULL;
char *token = NULL;
memcpy(copyStr, rawStatementsStr, sdslen(rawStatementsStr) + 1);

token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
while (token != NULL) {
    Condition *root;
    sds rawStatementStr = sdsnew(token);

    if (parseStatement(rawStatementStr, &root) == C_ERR) {
        serverLog(
                LL_WARNING,
                "[FILTER][FATAL] Stack condition parser failed, server wo
                rawStatementsStr);
        addReplyErrorFormat(
                c,
                "[FILTER][FATAL] Stack condition parser failed, server wo
                rawStatementsStr);
        return;
    }

    // serverLog(LL_DEBUG, "   ");
    // serverLog(LL_DEBUG, "[FILTER][PARSE] Condition Tree");
    // logCondition(root);

    Vector filteredMetakeys;
    vectorInit(&filteredMetakeys);
    for (size_t i = 0; i < vectorCount(&metakeys); ++i) {
        sds metakey = (sds) vectorGet(&metakeys, i);
        if (evaluateCondition(root, metakey)) {
            vectorAdd(&filteredMetakeys, metakey);
        }
    }
    vectorFree(&metakeys);
    metakeys = filteredMetakeys;

    sdsfree(rawStatementStr);
    freeConditions(root);

    token = strtok_r(NULL, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
}
```

metakeys - Parse & Evaluate Step

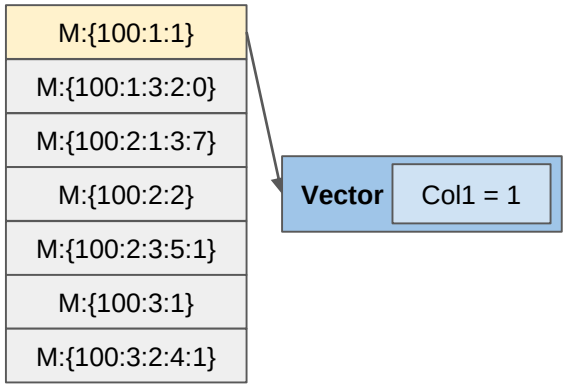
- evaluateCondition()
  - Statement Tree를 Traverse하며 해당 metakey가 조건에 맞는지 Evaluate
  - metakey는 여러 partition으로 이루어져 있음
    - ex) M:{100:2:3:5:1}
      → col2 = 3, col5 = 1
  - 단일 metakey에서 partition들을 parsing함

M:{100:1:1}
M:{100:1:3:2:0}
M:{100:2:1:3:7}
M:{100:2:2}
M:{100:2:3:5:1}
M:{100:3:1}
M:{100:3:2:4:1}

```
bool evaluateCondition(const Condition *root, const sds metakey) {
    MetaKeyInfo *metaKeyInfo = parseMetaKeyInfo(metakey);
    if (!metaKeyInfo->isPartitionString) {
        // TODO(totoro): Needs to handle 'isPartitionInt' case...
        return false;
    }

    Vector partitions;
    vectorInit(&partitions);
    if (_parsePartitions(metaKeyInfo->partitionInfo.partitionString,
                         &partitions) == C_ERR) {
        _freePartitionParameters(&partitions);
        return false;
    }

    // Prints a log of partitions.
    serverLog(LL_DEBUG, "[FILTER][EVALUATE] Parsed partitions");
    for (size_t i = 0; i < vectorCount(&partitions); ++i) {
        PartitionParameter *param = (PartitionParameter *) vectorGet(
                &partitions, i);
        if (param->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
            serverLog(LL_DEBUG,
                      "[FILTER][EVALUATE] columnId[%d] param sds value[%s]",
                      param->columnId, param->value.s);
        } else if (param->type == CONDITION_CHILD_VALUE_TYPE_LONG) {
            serverLog(LL_DEBUG,
                      "[FILTER][EVALUATE] columnId[%d] param long value[%ld]",
                      param->columnId, param->value.l);
        }
    }

    bool result = false;
    if (_evaluateCondition(root, &partitions)) {
        result = true;
    }

    _freePartitionParameters(&partitions);
    zfree(metaKeyInfo);
    return result;
}
```

metakeys - Parse & Evaluate Step

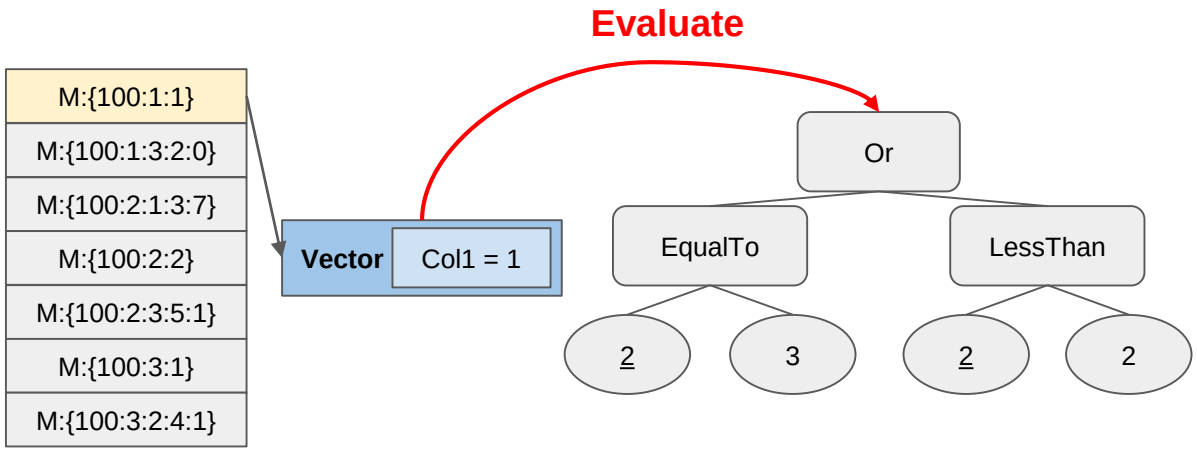
- evaluateCondition()
  - _parsePartitions()
    - Metakey의 partition들을 파싱하여 Vector(PartitionParameter)로 구성함
    - Partition Value는 String이 될 수 있으므로 sds type도 지원
    - ex) M:{100:1:1}




```c
int _parsePartitions(const char *partitionInfo, Vector *v) {
    char copyStr[MAX_TMPBUF_SIZE];
    char *savePtr = NULL;
    char *token = NULL;
    size_t size = strlen(partitionInfo) + 1;

    assert(size < MAX_TMPBUF_SIZE);
    memcpy(copyStr, partitionInfo, size);

    token = strtok_r(copyStr, RELMODEL_DELIMITER, &savePtr);
    while (savePtr[0] != '\0') {
        PartitionParameter *param = (PartitionParameter *) zmalloc(
                sizeof(PartitionParameter));
        param->columnId = atoi(token);
        if ((token = strtok_r(NULL, RELMODEL_DELIMITER, &savePtr)) == NULL) {
            zfree(param);
            serverLog(
                    LL_DEBUG,
                    "[FILTER][EVALUATE] PartitionInfo of key is invalid form:
                    partitionInfo);
            return C_ERR;
        }
        long valueLong;
        if (string2l(token, strlen(token), &valueLong) == 1) {
            param->type = CONDITION_CHILD_VALUE_TYPE_LONG;
            param->value.l = valueLong;
        } else {
            param->type = CONDITION_CHILD_VALUE_TYPE_SDS;
            param->value.s = sdsnew(token);
        }
        vectorAdd(v, param);
        token = strtok_r(NULL, RELMODEL_DELIMITER, &savePtr);
    }
    return C_OK;
}
```

metakeys - Parse & Evaluate Step

- evaluateCondition()
  - Recursive Function인 _evaluateCondition()을 사용하여 Statement Tree에 PartitionParameter들을 테스트 함

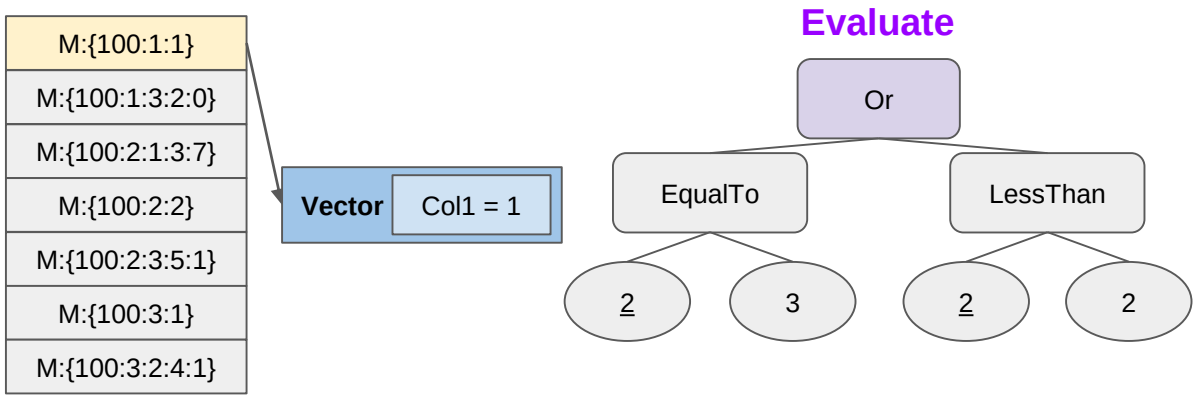



```
bool evaluateCondition(const Condition *root, const sds metakey) {
    MetaKeyInfo *metaKeyInfo = parseMetaKeyInfo(metakey);
    if (!metaKeyInfo->isPartitionString) {
        // TODO(totoro): Needs to handle 'isPartitionInt' case...
        return false;
    }

    Vector partitions;
    vectorInit(&partitions);
    if (_parsePartitions(metaKeyInfo->partitionInfo.partitionString,
                         &partitions) == C_ERR) {
        _freePartitionParameters(&partitions);
        return false;
    }

    // Prints a log of partitions.
    serverLog(LL_DEBUG, "[FILTER][EVALUATE] Parsed partitions");
    for (size_t i = 0; i < vectorCount(&partitions); ++i) {
        PartitionParameter *param = (PartitionParameter *) vectorGet(
                &partitions, i);
        if (param->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
            serverLog(LL_DEBUG,
                      "[FILTER][EVALUATE] columnId[%d] param sds value[%s]",
                      param->columnId, param->value.s);
        } else if (param->type == CONDITION_CHILD_VALUE_TYPE_LONG) {
            serverLog(LL_DEBUG,
                      "[FILTER][EVALUATE] columnId[%d] param long value[%ld]",
                      param->columnId, param->value.l);
        }
    }

    bool result = false;
    if (_evaluateCondition(root, &partitions)) {
        result = true;
    }

    _freePartitionParameters(&partitions);
    zfree(metaKeyInfo);
    return result;
}
```

metakeys - Parse & Evaluate Step

- _evaluateCondition()
  - Condition의 Type에 따라서 각기 다른 함수를 호출
    - Leaf → _evaluateLeafOperator()
    - Non-Leaf → _evaluateNonleafOperator()
  - Or는 Non-Leaf

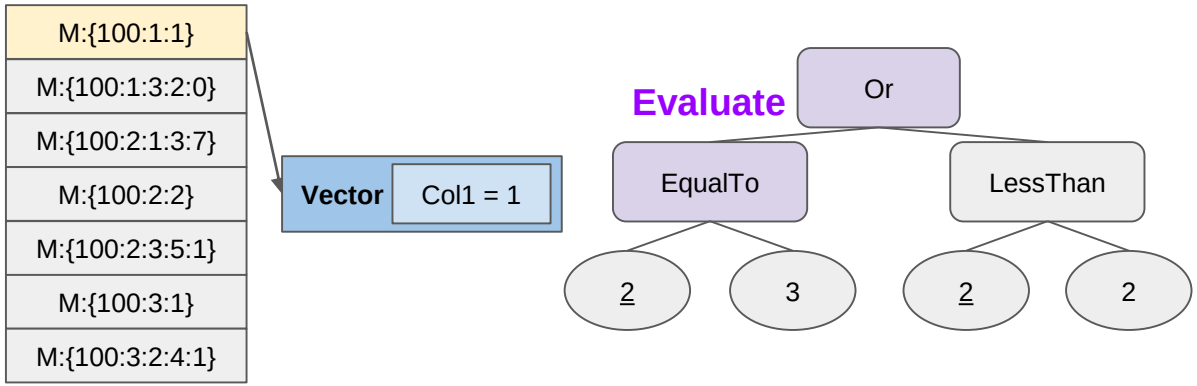
```
bool _evaluateCondition(const Condition *cond, Vector *partitions) {
    if (cond->isLeaf) {
        bool result = _evaluateLeafOperator(cond->op, cond->first, cond->second,
                                            partitions);
        serverLog(
                LL_DEBUG,
                "[FILTER][EVALUATE] _evaluateCondition leaf optype[%s], result[%
                _getOperatorStr(cond->op), result);
        return result;
    }

    bool result = _evaluateNonleafOperator(cond->op, cond->first, cond->second,
                                           partitions);
    serverLog(
            LL_DEBUG,
            "[FILTER][EVALUATE] _evaluateCondition non-leaf optype[%s], result[%
            _getOperatorStr(cond->op), result);
    return result;
}
```



metakeys - Parse & Evaluate Step

- _evaluateNonleafCondition()
  - Format에 맞지 않는 Error Checking
  - optype에 따라서 각기 다른 operator 호출
    - First, Second Child에 대해 각각 _evaluateCondition()을 Recursive Call

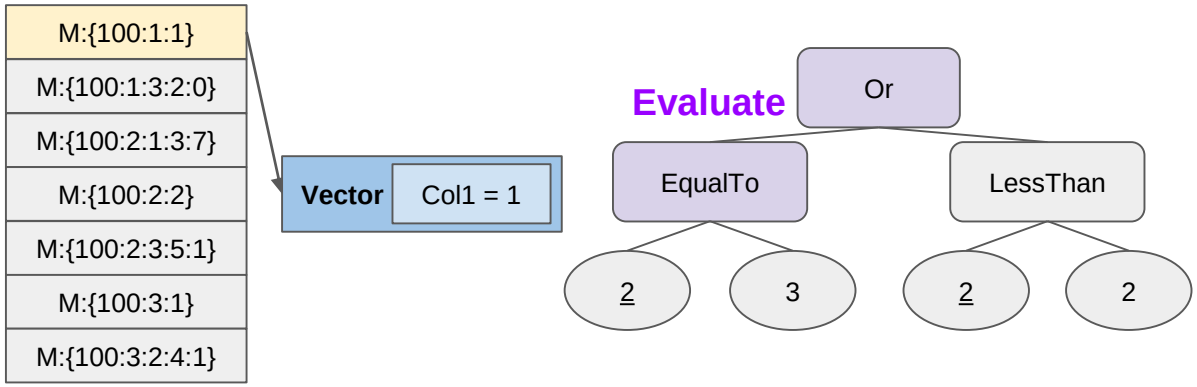
```
bool _evaluateNonleafOperator(const int optype, const ConditionChild *first,
                              const ConditionChild *second,
                              Vector *partitions) {
    if (
            optype != CONDITION_OP_TYPE_OR &&
            optype != CONDITION_OP_TYPE_AND &&
            optype != CONDITION_OP_TYPE_NOT
        ) {
        return false;
    }

    if (
            first == NULL ||
            first->type != CONDITION_CHILD_VALUE_TYPE_COND ||
            partitions == NULL
    ) {
        return false;
    }

    if(
            optype != CONDITION_OP_TYPE_NOT &&
            (
                second == NULL ||
                second->type != CONDITION_CHILD_VALUE_TYPE_COND
            )
    ) {
        return false;
    }

    if (optype == CONDITION_OP_TYPE_OR) {
        return _evaluateCondition(first->value.cond, partitions) ||
            _evaluateCondition(second->value.cond, partitions);
    } else if (optype == CONDITION_OP_TYPE_AND) {
        return _evaluateCondition(first->value.cond, partitions) &&
            _evaluateCondition(second->value.cond, partitions);
    } else if (optype == CONDITION_OP_TYPE_NOT) {
        return !_evaluateCondition(first->value.cond, partitions);
    } else {
        serverLog(
                LL_WARNING,
                "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                _getOperatorStr(optype));
        serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
    }

    return false;
}
```



metakeys - Parse & Evaluate Step

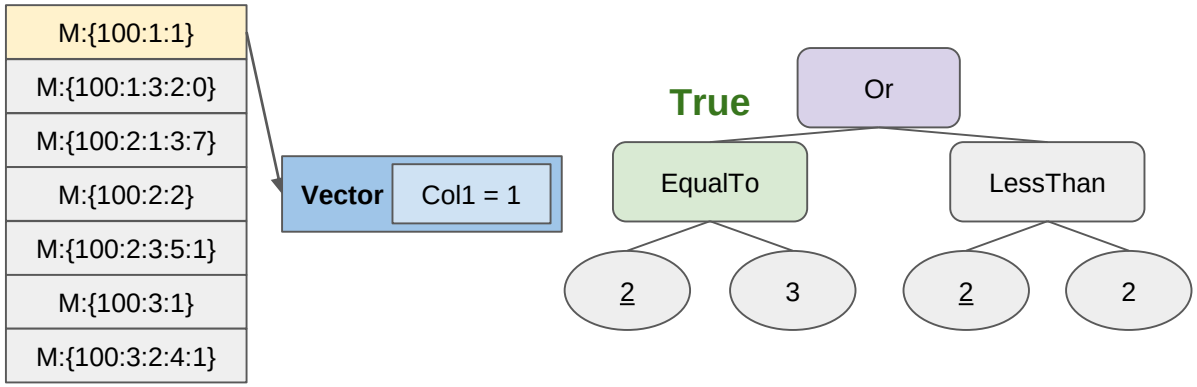
- _evaluateCondition()
  - EqualTo는 Leaf Condition이므로 _evaluateLeafCondition() 함수 호출

```
bool _evaluateCondition(const Condition *cond, Vector *partitions) {
    if (cond->isLeaf) {
        bool result = _evaluateLeafOperator(cond->op, cond->first, cond->second,
                                            partitions);
        serverLog(
                LL_DEBUG,
                "[FILTER][EVALUATE] _evaluateCondition leaf optype[%s], result[%
                _getOperatorStr(cond->op), result);
        return result;
    }

    bool result = _evaluateNonleafOperator(cond->op, cond->first, cond->second,
                                           partitions);
    serverLog(
            LL_DEBUG,
            "[FILTER][EVALUATE] _evaluateCondition non-leaf optype[%s], result[%
            _getOperatorStr(cond->op), result);
    return result;
}
```



metakeys - Parse & Evaluate Step

- _evaluateLeafCondition()
  - EqualTo에 대해서 Leaf Condition을 Check함
  - 첫번째 Child는 ColumnID
  - ColumnID가 parameter의 ColumnID와 같지 않으면 continue
  - EqualTo의 ColumnID가 2이므로, continue 후 True를 반환하게 됨

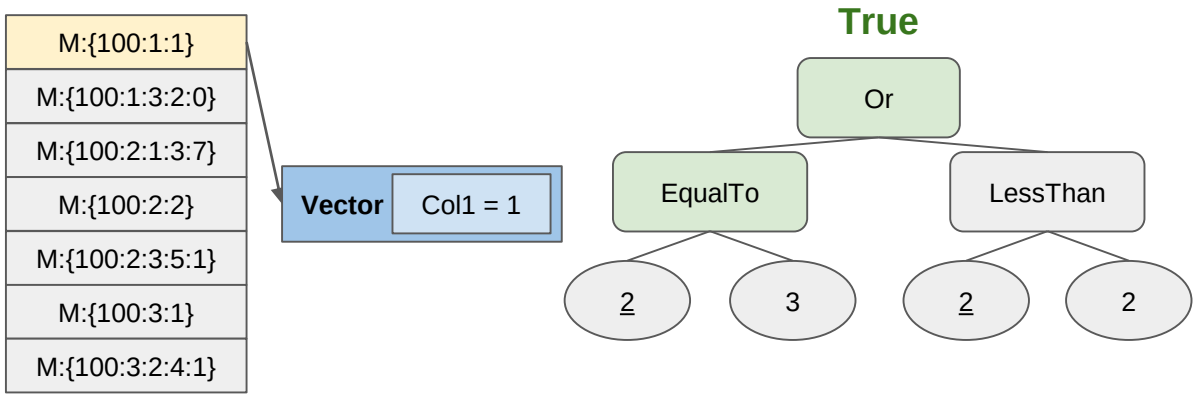



```
bool _evaluateLeafOperator(const int optype, const ConditionChild *first,
                           const ConditionChild *second, Vector *partitions) {
    for (size_t i = 0; i < vectorCount(partitions); ++i) {
        PartitionParameter *param = (PartitionParameter *) vectorGet(
                partitions, i);
        if (first == NULL || first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
            return false;
        }

        if (first->value.l != param->columnId) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_IS_NULL) {
            // TODO(totoro): Implements IS_NULL operator...
            return false;
        }

        if (optype == CONDITION_OP_TYPE_IS_NOT_NULL) {
            // TODO(totoro): Implements IS_NOT_NULL operator...
            return true;
        }

        if (
                second == NULL ||
                !(
                    second->type == CONDITION_CHILD_VALUE_TYPE_LONG ||
                    second->type == CONDITION_CHILD_VALUE_TYPE_SDS
                )
        ) {
            return false;
        }

        if (second->type != param->type) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_EQ) {
            if (second->type == CONDITION_CHILD_VALUE_TYPE_LONG) {
                return second->value.l == param->value.l;
            } else if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
                return sdscmp(second->value.s, param->value.s) == 0;
            }
        }

        if (
            optype == CONDITION_OP_TYPE_STRING_CONTAINS ||
            optype == CONDITION_OP_TYPE_STRING_ENDS_WITH ||
            optype == CONDITION_OP_TYPE_STRING_STARTS_WITH
        ) {
            if (second->type != CONDITION_CHILD_VALUE_TYPE_SDS) {
                return false;
            }

            sds converted = convertLikeStatementToGlobPattern(optype,
                                                               second->value.s);
            bool result = (bool) stringmatch(converted, param->value.s, 0);
            sdsfree(converted);
            return result;
        }

        if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
            return false;
        }

        if (optype == CONDITION_OP_TYPE_LT) {
            return param->value.l < second->value.l;
        } else if (optype == CONDITION_OP_TYPE_LTE) {
            return param->value.l <= second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GT) {
            return param->value.l > second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GTE) {
            return param->value.l >= second->value.l;
        } else {
            serverLog(
                    LL_WARNING,
                    "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                    _getOperatorStr(optype));
            serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
            return false;
        }
    }

    return true;
}
```

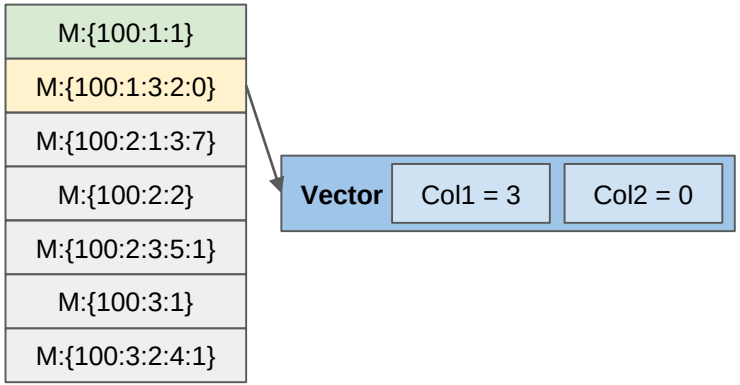
metakeys - Parse & Evaluate Step

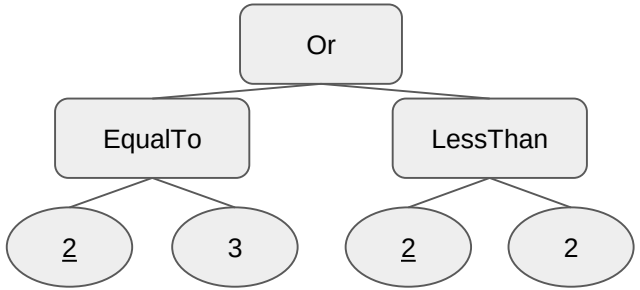
- _evaluateNonleafCondition()
  - Or 연산이므로, 전자의 연산이 True라면 후자의 연산은 계산되지 않고 바로 True를 리턴함

```
bool _evaluateNonleafOperator(const int optype, const ConditionChild *first,
                              const ConditionChild *second,
                              Vector *partitions) {
    if (
            optype != CONDITION_OP_TYPE_OR &&
            optype != CONDITION_OP_TYPE_AND &&
            optype != CONDITION_OP_TYPE_NOT
        ) {
        return false;
    }

    if (
            first == NULL ||
            first->type != CONDITION_CHILD_VALUE_TYPE_COND ||
            partitions == NULL
    ) {
        return false;
    }

    if(
            optype != CONDITION_OP_TYPE_NOT &&
            (
                second == NULL ||
                second->type != CONDITION_CHILD_VALUE_TYPE_COND
            )
    ) {
        return false;
    }

    if (optype == CONDITION_OP_TYPE_OR) {
        return _evaluateCondition(first->value.cond, partitions) ||
            _evaluateCondition(second->value.cond, partitions);
    } else if (optype == CONDITION_OP_TYPE_AND) {
        return _evaluateCondition(first->value.cond, partitions) &&
            _evaluateCondition(second->value.cond, partitions);
    } else if (optype == CONDITION_OP_TYPE_NOT) {
        return !_evaluateCondition(first->value.cond, partitions);
    } else {
        serverLog(
                LL_WARNING,
                "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                _getOperatorStr(optype));
        serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
    }

    return false;
}
```



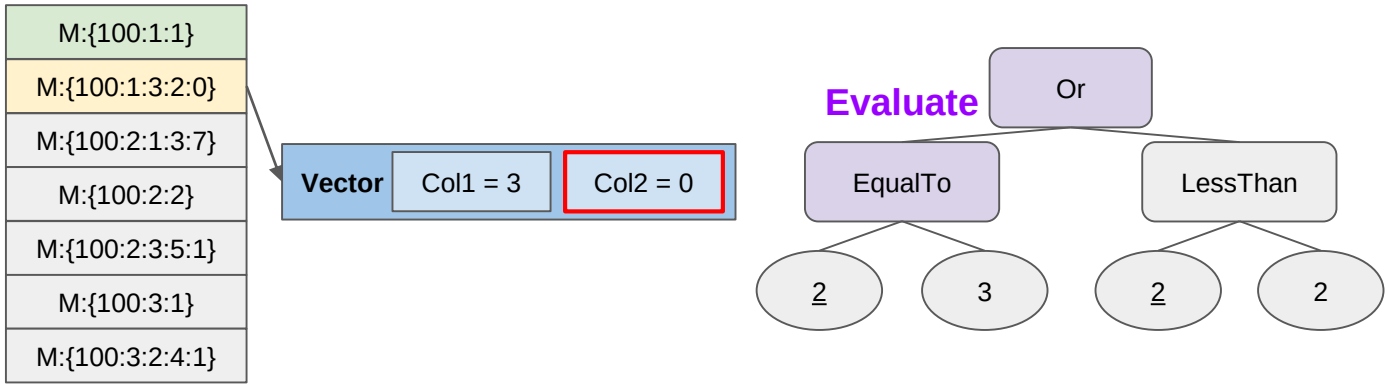
metakeys - Parse & Evaluate Step





metakeys - Parse & Evaluate Step

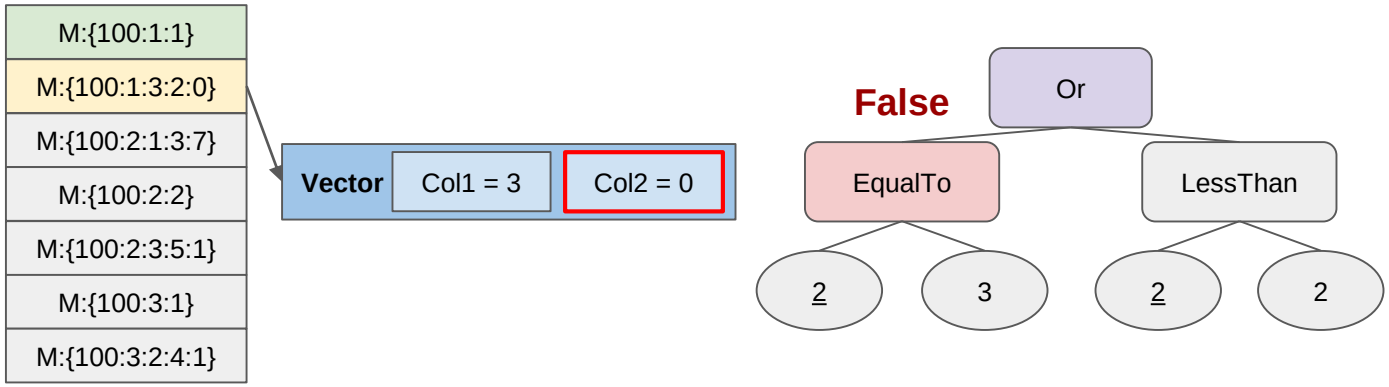
- _evaluateLeafCondition()
  - EqualTo에 대해서 Leaf Condition을 Check함
  - 첫번째 Child는 ColumnID
  - ColumnID가 parameter의 ColumnID와 같지 않으면 continue
  - 2번째 Partition을 검사하게 됨




```
bool _evaluateLeafOperator(const int optype, const ConditionChild *first,
                           const ConditionChild *second, Vector *partitions) {
    for (size_t i = 0; i < vectorCount(partitions); ++i) {
        PartitionParameter *param = (PartitionParameter *) vectorGet(
                partitions, i);
        if (first == NULL || first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
            return false;
        }

        if (first->value.l != param->columnId) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_IS_NULL) {
            // TODO(totoro): Implements IS_NULL operator...
            return false;
        }

        if (optype == CONDITION_OP_TYPE_IS_NOT_NULL) {
            // TODO(totoro): Implements IS_NOT_NULL operator...
            return true;
        }

        if (
                second == NULL ||
                !(
                    second->type == CONDITION_CHILD_VALUE_TYPE_LONG ||
                    second->type == CONDITION_CHILD_VALUE_TYPE_SDS
                )
        ) {
            return false;
        }

        if (second->type != param->type) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_EQ) {
            if (second->type == CONDITION_CHILD_VALUE_TYPE_LONG) {
                return second->value.l == param->value.l;
            } else if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
                return sdscmp(second->value.s, param->value.s) == 0;
            }
        }

        if (
            optype == CONDITION_OP_TYPE_STRING_CONTAINS ||
            optype == CONDITION_OP_TYPE_STRING_ENDS_WITH ||
            optype == CONDITION_OP_TYPE_STRING_STARTS_WITH
        ) {
            if (second->type != CONDITION_CHILD_VALUE_TYPE_SDS) {
                return false;
            }

            sds converted = convertLikeStatementToGlobPattern(optype,
                                                               second->value.s);
            bool result = (bool) stringmatch(converted, param->value.s, 0);
            sdsfree(converted);
            return result;
        }

        if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
            return false;
        }

        if (optype == CONDITION_OP_TYPE_LT) {
            return param->value.l < second->value.l;
        } else if (optype == CONDITION_OP_TYPE_LTE) {
            return param->value.l <= second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GT) {
            return param->value.l > second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GTE) {
            return param->value.l >= second->value.l;
        } else {
            serverLog(
                    LL_WARNING,
                    "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                    _getOperatorStr(optype));
            serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
            return false;
        }
    }

    return true;
}
```

metakeys - Parse & Evaluate Step

- _evaluateLeafCondition()
  - EqualTo의 경우는, Long은 ==, sds는 sdscmp()로 비교함
  - 0 != 3 이므로 False를 반환

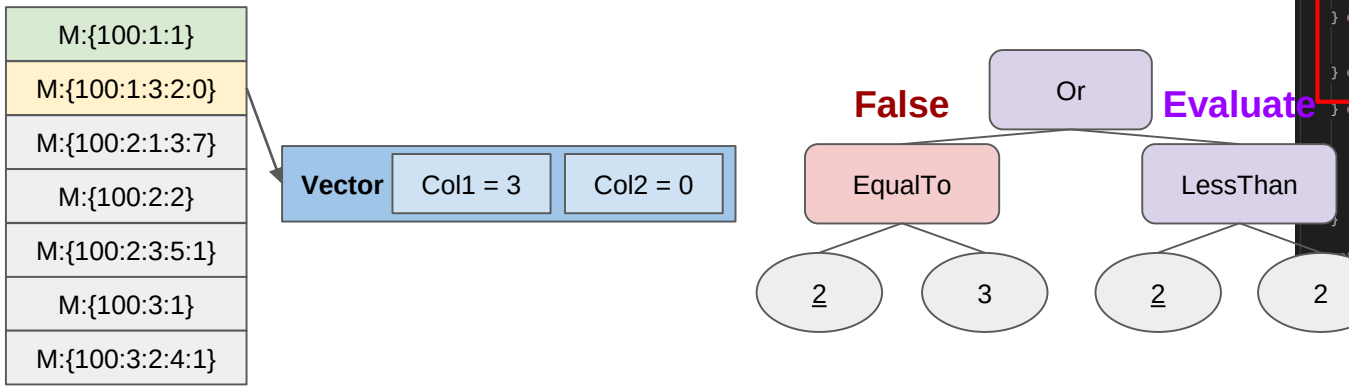



```
bool _evaluateLeafOperator(const int optype, const ConditionChild *first,
                           const ConditionChild *second, Vector *partitions) {
    for (size_t i = 0; i < vectorCount(partitions); ++i) {
        PartitionParameter *param = (PartitionParameter *) vectorGet(
                partitions, i);
        if (first == NULL || first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
            return false;
        }

        if (first->value.l != param->columnId) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_IS_NULL) {
            // TODO(totoro): Implements IS_NULL operator...
            return false;
        }

        if (optype == CONDITION_OP_TYPE_IS_NOT_NULL) {
            // TODO(totoro): Implements IS_NOT_NULL operator...
            return true;
        }

        if (
                second == NULL ||
                !(
                    second->type == CONDITION_CHILD_VALUE_TYPE_LONG ||
                    second->type == CONDITION_CHILD_VALUE_TYPE_SDS
                )
        ) {
            return false;
        }

        if (second->type != param->type) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_EQ) {
            if (second->type == CONDITION_CHILD_VALUE_TYPE_LONG) {
                return second->value.l == param->value.l;
            } else if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
                return sdscmp(second->value.s, param->value.s) == 0;
            }
        }

        if (
            optype == CONDITION_OP_TYPE_STRING_CONTAINS ||
            optype == CONDITION_OP_TYPE_STRING_ENDS_WITH ||
            optype == CONDITION_OP_TYPE_STRING_STARTS_WITH
        ) {
            if (second->type != CONDITION_CHILD_VALUE_TYPE_SDS) {
                return false;
            }

            sds converted = convertLikeStatementToGlobPattern(optype,
                                                               second->value.s);
            bool result = (bool) stringmatch(converted, param->value.s, 0);
            sdsfree(converted);
            return result;
        }

        if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
            return false;
        }

        if (optype == CONDITION_OP_TYPE_LT) {
            return param->value.l < second->value.l;
        } else if (optype == CONDITION_OP_TYPE_LTE) {
            return param->value.l <= second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GT) {
            return param->value.l > second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GTE) {
            return param->value.l >= second->value.l;
        } else {
            serverLog(
                    LL_WARNING,
                    "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                    _getOperatorStr(optype));
            serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
            return false;
        }
    }

    return true;
}
```

metakeys - Parse & Evaluate Step

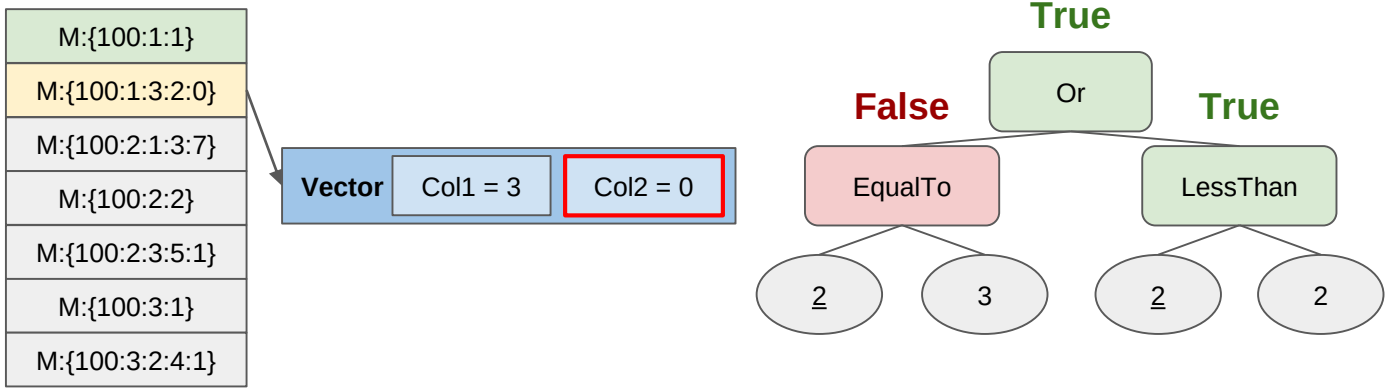
- _evaluateNonleafCondition()
  - Or 연산이므로, 전자의 연산이 False라면 후자의 연산도 계산해야함

```
bool _evaluateNonleafOperator(const int optype, const ConditionChild *first,
                              const ConditionChild *second,
                              Vector *partitions) {
    if (
            optype != CONDITION_OP_TYPE_OR &&
            optype != CONDITION_OP_TYPE_AND &&
            optype != CONDITION_OP_TYPE_NOT
        ) {
        return false;
    }

    if (
            first == NULL ||
            first->type != CONDITION_CHILD_VALUE_TYPE_COND ||
            partitions == NULL
    ) {
        return false;
    }

    if(
            optype != CONDITION_OP_TYPE_NOT &&
            (
                second == NULL ||
                second->type != CONDITION_CHILD_VALUE_TYPE_COND
            )
    ) {
        return false;
    }

    if (optype == CONDITION_OP_TYPE_OR) {
        return _evaluateCondition(first->value.cond, partitions) ||
            _evaluateCondition(second->value.cond, partitions);
    } else if (optype == CONDITION_OP_TYPE_AND) {
        return _evaluateCondition(first->value.cond, partitions) &&
            _evaluateCondition(second->value.cond, partitions);
    } else if (optype == CONDITION_OP_TYPE_NOT) {
        return !_evaluateCondition(first->value.cond, partitions);
    } else {
        serverLog(
                LL_WARNING,
                "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                _getOperatorStr(optype));
        serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
    }

    return false;
```



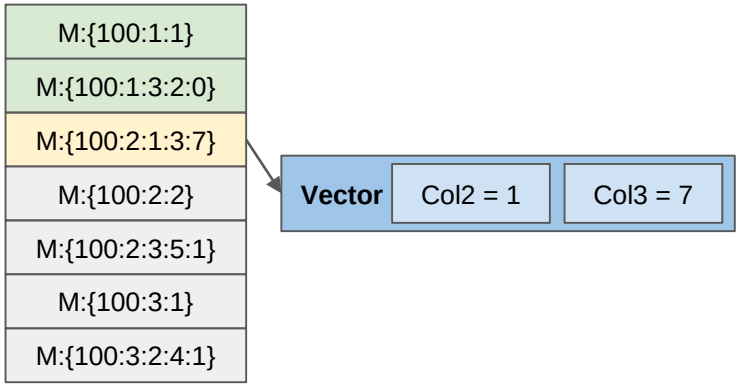
metakeys - Parse & Evaluate Step

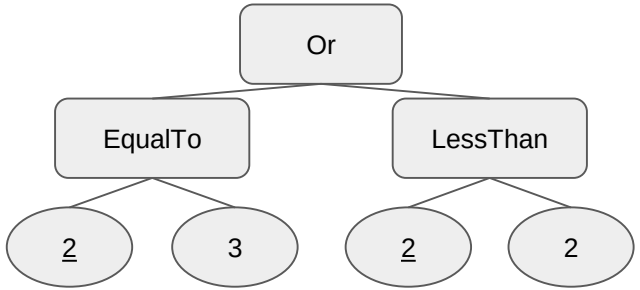
- _evaluateLeafCondition()
  - LessThan(Equal), GreaterThan(Equal)의 경우는, Long Type일 경우에만 계산함
  - 0 < 2 이므로 True를 반환
  - 이 경우, Or 연산도 True




```
bool _evaluateLeafOperator(const int optype, const ConditionChild *first,
                           const ConditionChild *second, Vector *partitions) {
    for (size_t i = 0; i < vectorCount(partitions); ++i) {
        PartitionParameter *param = (PartitionParameter *) vectorGet(
                partitions, i);
        if (first == NULL || first->type != CONDITION_CHILD_VALUE_TYPE_LONG) {
            return false;
        }

        if (first->value.l != param->columnId) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_IS_NULL) {
            // TODO(totoro): Implements IS_NULL operator...
            return false;
        }

        if (optype == CONDITION_OP_TYPE_IS_NOT_NULL) {
            // TODO(totoro): Implements IS_NOT_NULL operator...
            return true;
        }

        if (
                second == NULL ||
                !(
                    second->type == CONDITION_CHILD_VALUE_TYPE_LONG ||
                    second->type == CONDITION_CHILD_VALUE_TYPE_SDS
                )
        ) {
            return false;
        }

        if (second->type != param->type) {
            continue;
        }

        if (optype == CONDITION_OP_TYPE_EQ) {
            if (second->type == CONDITION_CHILD_VALUE_TYPE_LONG) {
                return second->value.l == param->value.l;
            } else if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
                return sdscmp(second->value.s, param->value.s) == 0;
            }
        }

        if (
            optype == CONDITION_OP_TYPE_STRING_CONTAINS ||
            optype == CONDITION_OP_TYPE_STRING_ENDS_WITH ||
            optype == CONDITION_OP_TYPE_STRING_STARTS_WITH
        ) {
            if (second->type != CONDITION_CHILD_VALUE_TYPE_SDS) {
                return false;
            }

            sds converted = convertLikeStatementToGlobPattern(optype,
                                                               second->value.s);
            bool result = (bool) stringmatch(converted, param->value.s, 0);
            sdsfree(converted);
            return result;
        }

        if (second->type == CONDITION_CHILD_VALUE_TYPE_SDS) {
            return false;
        }

        if (optype == CONDITION_OP_TYPE_LT) {
            return param->value.l < second->value.l;
        } else if (optype == CONDITION_OP_TYPE_LTE) {
            return param->value.l <= second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GT) {
            return param->value.l > second->value.l;
        } else if (optype == CONDITION_OP_TYPE_GTE) {
            return param->value.l >= second->value.l;
        } else {
            serverLog(
                    LL_WARNING,
                    "[FILTER][EVALUATE][FATAL] Invalid operator type: %s",
                    _getOperatorStr(optype));
            serverPanic("[FILTER][EVALUATE][FATAL] Invalid operator type");
            return false;
        }
    }

    return true;
}
```

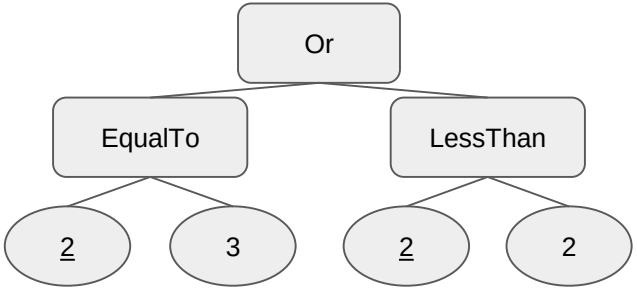
metakeys - Parse & Evaluate Step





metakeys - Parse & Evaluate Step

M:{100:1:1}
M:{100:1:3:**2:0**}
M:{100:**2:1**:3:7}
M:{100:**2:2**}
M:{100:**2:3**:5:1}
M:{100:3:1}
M:{100:3:2:4:1}



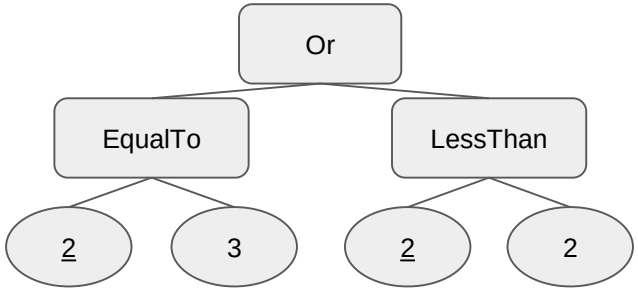
metakeys - Example

- ex) redis-cli> metakeys M:{100:*} 3*2*EqualTo:2*2*LessThan:Or:$

MetaDict

M:{10:2:3}	hashdict
M:{100:1:1}	hashdict
M:{100:1:3:2:0}	hashdict
M:{100:2:1:3:7}	hashdict
M:{100:2:2}	hashdict
M:{100:2:3:5:1}	hashdict
M:{100:3:1}	hashdict
M:{100:3:2:4:1}	hashdict
M:{200:2:1}	hashdict
M:{200:2:3}	hashdict

Evaluation Tree



Results

M:{100:1:1}
M:{100:1:3:2:0}
M:{100:2:1:3:7}
M:{100:2:2}

M:{100:2:3:5:1}
M:{100:3:1}
M:{100:3:2:4:1}

metakeys - Code Structure

metakeysCommand()

Initialize Step

- MetaDict에서 MetaKey들을 Collect
- Pattern matching

Collect

Validate Step

- Stringfied Statements가 Valid Form인지 Check (미구현)

~~**Validate**~~

Parse & Evaluate Step

- Stringfied Statement를 Evaluation Tree로 Parse
- Evaluation Tree에 각 Metakey를 Evaluate, 조건에 맞으면 유지, 조건에 맞지 않으면 Filter

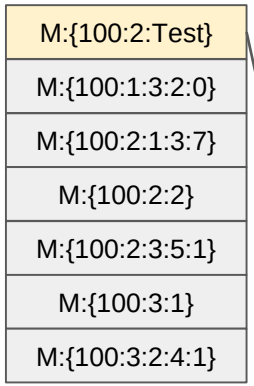
Parse
Evaluate
Filter

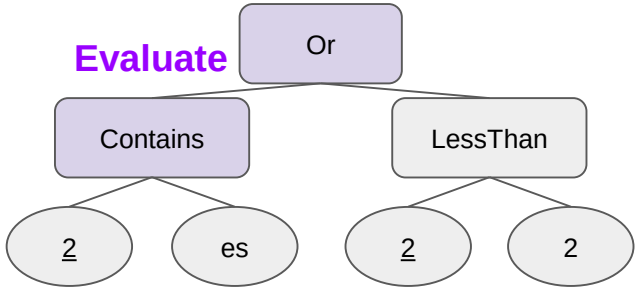
```
void metakeysCommand(client *c){
    /*Parses stringfied stack structure to readable parameters*/
    sds pattern = (sds) c->argv[1]->ptr;
    bool allkeys = (pattern[0] == '*' && pattern[1] == '\0');

    /*Initializes Vector for Metadict keys*/
    Vector metakeys;
    vectorTypeInit(&metakeys, STL_TYPE_SDS);

    /*Pattern match searching for metakeys*/
    ...

    /* If rawStatements is null or empty, prints pattern matching
     * results only...
     */
    ...

    sds rawStatementsStr = (sds) c->argv[2]->ptr;

    if (!validateStatements(rawStatementsStr)) {
        serverLog(LL_WARNING, "[FILTER] Stack structure is not valid form:
                  rawStatementsStr);
        addReplyErrorFormat(c, "[FILTER] Stack structure is not valid form:
                            rawStatementsStr);
        return;
    }

    /*Search each statmenet*/
    token = strtok_r(copyStr, PARTITION_FILTER_STATEMENT_SUFFIX, &savePtr);
    while (token != NULL) {
        // Parse
        ...

        // Evaluate
        ...

        // Filter
        ...
    }

    /*Prints out target partitions*/
    /*Scan data to client*/
    _addReplyMetakeysResults(c, &metakeys);
    vectorFree(&metakeys);
}
```

metakeys - Parse & Evaluate Step

- _evaluateLeafCondition()
  - 만약 연산이 LIKE Query라면?
  - Like의 parameter를 연산에 맞게 Glob Pattern으로 수정
    - CONTAINS → *Pattern*
    - ENDS_WITH → *Pattern
    - STARTS_WITH → Pattern*
  - sds 타입에 대해서만 테스트

```
sds convertLikeStatementToGlobPattern(const int optype,
                                      const sds likeStatement) {
    if (optype == CONDITION_OP_TYPE_STRING_CONTAINS) {
        return sdscatfmt(sdsempty(), "%s%S%s", "*", likeStatement, "*");
    } else if (optype == CONDITION_OP_TYPE_STRING_ENDS_WITH) {
        return sdscatfmt(sdsempty(), "%s%S", "*", likeStatement);
    } else if (optype == CONDITION_OP_TYPE_STRING_STARTS_WITH) {
        return sdscatfmt(sdsempty(), "%S%s", likeStatement, "*");
    } else {
        return NULL;
    }
}
```

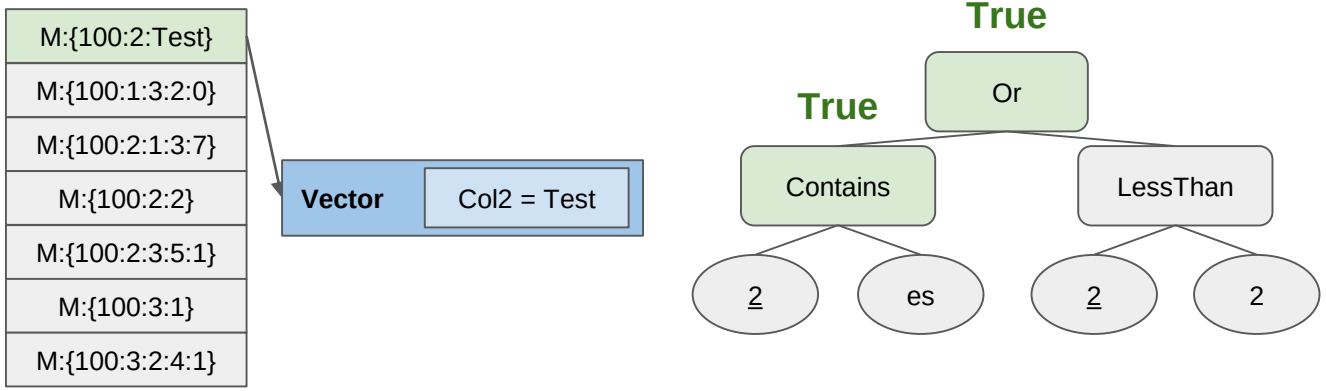




metakeys - Parse & Evaluate Step

- _evaluateLeafCondition()
  - Test는 *es* Pattern에 적합하므로 true를 반환함

```
sds convertLikeStatementToGlobPattern(const int optype,
                                      const sds likeStatement) {
    if (optype == CONDITION_OP_TYPE_STRING_CONTAINS) {
        return sdscatfmt(sdsempty(), "%s%S%s", "*", likeStatement, "*");
    } else if (optype == CONDITION_OP_TYPE_STRING_ENDS_WITH) {
        return sdscatfmt(sdsempty(), "%s%S", "*", likeStatement);
    } else if (optype == CONDITION_OP_TYPE_STRING_STARTS_WITH) {
        return sdscatfmt(sdsempty(), "%S%s", likeStatement, "*");
    } else {
        return NULL;
    }
}
```



Summary

- metakeys
  - metakeys Command
  - metakeys Command Flow
    - Parse
      - Parse to Statement Tree by using “Stack”
      - Statement Tree
    - Evaluate & Filter
      - Acceptation for unmatched partition concept

Q & A