Comparison Script 수정

옵션 --includeDbs --includeCollections 추가

기존 excludeDbs , excludeCollections 에서 새로운 옵션 추가로 다양한 방법으로 시도할 수 있다.

또한 include/exclude Collections 옵션에서 새로운 입력 방식을 추가하여 사용자 편의를 더했다.

기존 collection 이름만 입력하는 방식에서 'DB 이름/Collection 이름' 형식으로 입력해야한다.

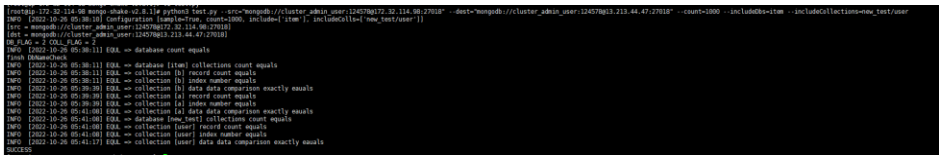
같은 이름의 collection 이라도 DB 가 다르면 다른 collection 이기 때문에 이를 Script 에서 정확하게 입력받을 수 있게끔 하였다.

만약 Collection 이름이나 DB 이름에 '/' 문자가 들어간다면 오류가 발생할 수 있다 주의 하자

간단하게 테스트 클러스터를 구성해보았다. src 데이터베이스와 dst 데이터베이스의 구조는 동일하다.

--includeDbs =item --includeCollections=new_test/user

item DB 의 모든 COLLECTION 을 비교 하고, new_test DB 의 user 컬렉션 만 추가로 비교해 보는 것



테스트 결과를 보면 item DB 의 a,b collection 과 new_test DB 의 user collection 에서만 비교를 진행하였다. (config 와 같은 기반 DB 에 대한 것은 테스트 환경에서는 신경쓰지 않아도 된다.)

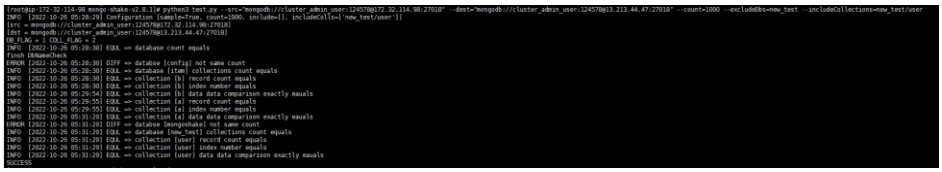
--includeDbs =new_test --excludeCollections=new_test/item

new_test 디비에서 item 컬렉션을 제외한 컬렉션을 비교하게 될것이다.

마찬가지로 테스트 결과를 보면 new_test 에서 item 컬렉션을 제외한 user 컬렉션만 비교된 것을 볼 수 있다.

--excludeDbs =new_test --includeCollections=new_test/user

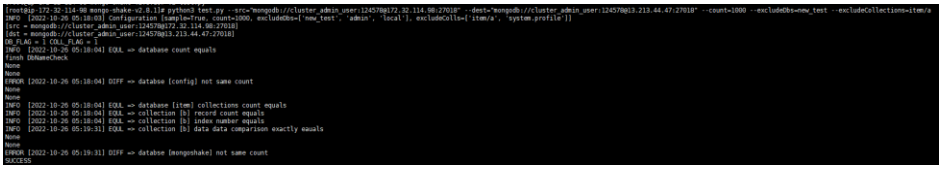
new_test DB 를 제외하고 비교하지만 new_test 내부의 user 컬렉션은 비교에 포함시키고 싶을때



테스트 결과를 보면 new_test 를 제외한 모든 DB 를 확인함과 동시에 제외된 new_test 에서 user 만 비교를 더 한다.

--excludeDbs =new_test --excludeCollections=item/a

new_test DB 를 제외하고 나머지 모든 DB 를 비교하지만 item DB 의 a 컬렉션만 비교에서 제외시킨다.



테스트 결과 성공

```
#!/usr/bin/env python

# -*- coding:utf-8 -*-

from lib2to3.pgen2.token import EQUAL
from shelve import DbfilenameShelf
import pymongo
import time
import random
import sys
```

```
import getopt

# constant
DB_FLAG = 0             # 1: EXCLUDE 2 : INCLUDE :
COLL_FLAG = 0

COMPARISION_COUNT = "comparison_count"
COMPARISION_MODE = "comparisonMode"
EXCLUDE_DBS = "excludeDbs"
INCLUDE_DBS = "includeDbs"
EXCLUDE_COLLS = "excludeColls"
INCLUDE_COLLS = "includeColls"
SAMPLE = "sample"
# we don't check collections and index here because sharding's
collection(`db.stats`) is splitted.
CheckList = {"objects": 1, "numExtents": 1, "ok": 1}
configure = {}

def log_info(message):
    print("INFO  [%s] %s " % (time.strftime('%Y-%m-%d %H:%M:%S'), message))

def log_error(message):
    print("ERROR [%s] %s " % (time.strftime('%Y-%m-%d %H:%M:%S'), message))

class MongoCluster:

    # pymongo connection
    conn = None

    # connection string
    url = ""

    def __init__(self, url):
        self.url = url

    def connect(self):
        self.conn = pymongo.MongoClient(self.url)

    def close(self):
        self.conn.close()
```

```
def filter_check(m):
    new_m = {}
    for k in CheckList:
        new_m[k] = m[k]
    return new_m

# check database names and collections
def CollCheck(db,srcColls,dstColls,srcDb, dstDb):

    # if len(srcColls) != len(dstColls):
    #     log_error("DIFF => database [%s] collections count not equals,
src[%s], dst[%s]" % (db, srcColls, dstColls))
    #     return False
    # else:
    log_info("EQUL => database [%s] collections count equals" % (db))

    for coll in srcColls:
        # if coll in configure[EXCLUDE_COLLS]:
        #     log_info("IGNR => ignore collection [%s]" % coll)
        #     continue

        if dstColls.count(coll) == 0:
            log_error("DIFF => collection only in source [%s]" % (coll))
            return False

        srcColl = srcDb[coll]
        dstColl = dstDb[coll]
        # comparison collection records number
        if srcColl.count_documents({}) != dstColl.count_documents({}):
        # count() is deprecated in 3.7 . Use Collection.count_documents
instead.
            log_error("DIFF => collection [%s] record count not equals" %
(coll))
            return False
        else:
            log_info("EQUL => collection [%s] record count equals" % (coll))

        # comparison collection index number
        src_index_length = len(srcColl.index_information())
        dst_index_length = len(dstColl.index_information())
```

```
        if src_index_length != dst_index_length:
            log_error("DIFF => collection [%s] index number not equals:
src[%r], dst[%r]" % (coll, src_index_length, dst_index_length))
            return False
        else:
            log_info("EQUL => collection [%s] index number equals" % (coll))

        # check sample data
        if not data_comparison(srcColl, dstColl, configure[COMPARISION_MODE]):
            log_error("DIFF => collection [%s] data comparison not equals" %
(coll))
            return False
        else:
            log_info("EQUL => collection [%s] data data comparison exactly
eauals" % (coll))

def DbNameCheck(srcDbNames,dstDbNames) :
    if len(srcDbNames) != len(dstDbNames):
        log_error("DIFF => database count not equals src[%s] !=
dst[%s].₩nsrc: %s₩ndst: %s" % (len(srcDbNames),

len(dstDbNames),

srcDbNames,

dstDbNames))
        return False
    else:
        log_info("EQUL => database count equals")


"""
    check meta data. include db.collection names and stats()
"""
def check(src, dst):
    #
    # check metadata
    #
    srcDbNames = src.conn.list_database_names()
    #The database_names is deprecated from 3.7 onwards and been replaced by
list_database_names()
```

```
    dstDbNames = dst.conn.list_database_names()


    if DB_FLAG ==1 :                  # excludeDB exist
        srcDbNames = [db for db in srcDbNames if db not in
configure[EXCLUDE_DBS]]
        dstDbNames = [db for db in dstDbNames if db not in
configure[EXCLUDE_DBS]]

        DbNameCheck(srcDbNames,dstDbNames)
        print("finsh DbNameCheck")
        if COLL_FLAG == 1 :          # excludeCollection exist
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False


                # db.stats() comparison
                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                srcColls = [coll for coll in srcColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]
                dstColls = [coll for coll in dstColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]
                print(srcColls.sort())
                print(dstColls.sort())
                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue


                CollCheck(db,srcColls,dstColls,srcDb, dstDb)
        elif COLL_FLAG == 2 :       # includeCollection exist
            for db in srcDbNames:
```

```
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False

                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue

                CollCheck(db,srcColls,dstColls,srcDb, dstDb)

            for COLL in configure[INCLUDE_COLLS] :
                tmpDB,tmpCOLL = COLL.split('/')

CollCheck(tmpDB,[tmpCOLL],[tmpCOLL],src.conn[tmpDB],dst.conn[tmpDB])
        else :
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False

                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue

                CollCheck(db,srcColls,dstColls,srcDb, dstDb)
```

```
    elif DB_FLAG ==2 :                  # includeDB exist
        srcDbNames = [db for db in srcDbNames if db in configure[INCLUDE_DBS]]
        dstDbNames = [db for db in dstDbNames if db in configure[INCLUDE_DBS]]

        DbNameCheck(srcDbNames,dstDbNames)
        print("finsh DbNameCheck")
        if COLL_FLAG == 1 :         # excludeCollection exist
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False


                # db.stats() comparison
                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                srcColls = [coll for coll in srcColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]
                dstColls = [coll for coll in dstColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue


                CollCheck(db,srcColls,dstColls,srcDb, dstDb)
        elif COLL_FLAG == 2 :       # includeCollection exist
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False

                srcDb = src.conn[db]
                dstDb = dst.conn[db]
```

```
                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue

                CollCheck(db,srcColls,dstColls,srcDb, dstDb)

            for COLL in configure[INCLUDE_COLLS] :
                tmpDB,tmpCOLL = COLL.split('/')

CollCheck(tmpDB,[tmpCOLL],[tmpCOLL],src.conn[tmpDB],dst.conn[tmpDB])
        else :
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False

                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                srcColls = [coll for coll in srcColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]
                dstColls = [coll for coll in dstColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue

                CollCheck(db,srcColls,dstColls,srcDb, dstDb)
    else :
```

```
        srcDbNames = [db for db in srcDbNames if db not in
configure[EXCLUDE_DBS]]
        dstDbNames = [db for db in dstDbNames if db not in
configure[EXCLUDE_DBS]]
        DbNameCheck(srcDbNames,dstDbNames)
        if COLL_FLAG == 1 :         # excludeCollection exist
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False


                # db.stats() comparison
                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                srcColls = [coll for coll in srcColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]
                dstColls = [coll for coll in dstColls if
''.join(db)+'/'+''.join(coll) not in configure[EXCLUDE_COLLS] ]

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue


                CollCheck(db,srcColls,dstColls,srcDb, dstDb)
        elif COLL_FLAG == 2 :       # includeCollection exist
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False

                srcDb = src.conn[db]
                dstDb = dst.conn[db]
```

```
                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue

                CollCheck(db,srcColls,dstColls,srcDb, dstDb)

            for COLL in configure[INCLUDE_COLLS] :
                tmpDB,tmpCOLL = COLL.split('/')

CollCheck(tmpDB,[tmpCOLL],[tmpCOLL],src.conn[tmpDB],dst.conn[tmpDB])
        else :
            for db in srcDbNames:
                if dstDbNames.count(db) == 0:
                    log_error("DIFF => database [%s] only in srcDb" % (db))
                    return False

                srcDb = src.conn[db]
                dstDb = dst.conn[db]

                srcColls = srcDb.list_collection_names()
                # The collection_names is deprecated from 3.7 onwards and been
replaced by list_collection_names()
                dstColls = dstDb.list_collection_names()

                if len(srcColls) - len(dstColls) is not 0 :
                    log_error("DIFF => databse [%s] not same count " % (db))
                    continue

                CollCheck(db,srcColls,dstColls,srcDb, dstDb)
    return True



"""
    check sample data. comparison every entry
"""
```

```
def data_comparison(srcColl, dstColl, mode):
    if mode == "no":
        return True
    elif mode == "sample":
        # srcColl.count() mus::t equals to dstColl.count()
        count = configure[COMPARISION_COUNT] if configure[COMPARISION_COUNT]
<= srcColl.count_documents({}) else srcColl.count_documents({})
    else: # all
        count = srcColl.count_documents({})

    if count == 0:
        return True

    rec_count = count
    batch = 16
    show_progress = (batch * 64)
    total = 0
    while count > 0:
        # sample a bounch of docs

        docs = srcColl.aggregate([{"$sample": {"size":batch}}])
        while docs.alive:
            doc = docs.next()
            migrated = dstColl.find_one(doc["_id"])
            # both origin and migrated bson is Map . so use ==
            if doc != migrated:
                log_error("DIFF => src_record[%s], dst_record[%s]" % (doc,
migrated))
                return False

        total += batch
        count -= batch

        if total % show_progress == 0:
            log_info("  ... process %d docs, %.2f %% !" % (total, total *
100.0 / rec_count))


    return True
```

```
def usage():
    print('|--------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------|')
    print("| Usage_with --exclude: ./comparison.py --src=localhost:27017/db? -
-dest=localhost:27018/db? --count=10000 (the sample number) --
excludeDbs=admin,local --excludeCollections=system.profile --
comparisonMode=sample/all/no (sample: comparison sample number, default; all:
comparison all data; no: only comparison outline without data)  |")
    print('|--------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------|')
    print("| Usage_with --include: ./comparison.py --src=localhost:27017/db? -
-dest=localhost:27018/db? --count=10000 (the sample number) --
includeDbs=test,itmes ( sample dbs )  --includeCollections=test.users --
comparisonMode=sample/all/no (sample: comparison sample number, default; all:
comparison all data; no: only comparison outline without data)  |")
    print('|--------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------|')
    print('| Like : ./comparison.py --src="localhost:3001" --
dest=localhost:3100  --count=1000  --excludeDbs=admin,local,mongoshake --
excludeCollections=system.profile --comparisonMode=sample  |')
    print('|--------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------|')
    exit(0)

if __name__ == "__main__":
    opts, args = getopt.getopt(sys.argv[1:], "hs:d:n:e:x:i:n", ["help",
"src=", "dest=", "count=", "excludeDbs=", "excludeCollections=",
"comparisonMode=","includeDbs=","includeCollections="])

    configure[SAMPLE] = True
    configure[EXCLUDE_DBS] = []
    configure[EXCLUDE_COLLS] = []
    configure[INCLUDE_DBS]=[]
    configure[INCLUDE_COLLS]=[]

    srcUrl, dstUrl = "", ""
```

```
    for key, value in opts:
        if key in ("-h", "--help"):
            usage()
        if key in ("-s", "--src"):
            srcUrl = value
        if key in ("-d", "--dest"):
            dstUrl = value
        if key in ("-n", "--count"):
            configure[COMPARISION_COUNT] = int(value)
        if key in ("-e", "--excludeDbs"):
            if key in ("-i","--includeDbs"):                                    #
To check if include option exist
                log_info("Cant use includeDbs / excludeDbs in same time")
#
                exit(1)
            DB_FLAG = 1#
            configure[EXCLUDE_DBS] = value.split(",")
        elif key in ("-i","--includeDbs"):
            if key in ("-e","--excludeDbs"):                                    #
To check if include option exist
                log_info("Cant use includeDbs / excludeDbs in same time")
#
                exit(1)          #
            DB_FLAG = 2
            configure[INCLUDE_DBS] = value.split(",")                           #
        if key in ("-x", "--excludeCollections"):
            if key in ("-n","--includeCollections"):
#
                log_info("Cant use includeColl / excludeColl in same time")
#
                exit(1)
            COLL_FLAG = 1
#
            configure[EXCLUDE_COLLS] = value.split(",")
        elif key in ("-n","--includeCollections"):                              #
            if key in ("-x","--excludeCollections"):
#
                log_info("Cant use includeColl / excludeColl in same time")
#
                exit(1)
            COLL_FLAG = 2
```

```
            configure[INCLUDE_COLLS] = value.split(",")                    #

        if key in ("--comparisonMode"):
            print(value)
            if value != "all" and value != "no" and value != "sample":
                log_info("comparisonMode[%r] illegal" % (value))
                exit(1)
            configure[COMPARISION_MODE] = value
    if COMPARISION_MODE not in configure:
        configure[COMPARISION_MODE] = "sample"

    # params verify
    if len(srcUrl) == 0 or len(dstUrl) == 0:
        usage()

    # default count is 10000
    if configure.get(COMPARISION_COUNT) is None or
configure.get(COMPARISION_COUNT) <= 0:
        configure[COMPARISION_COUNT] = 10000

    # ignore databases
    configure[EXCLUDE_DBS] += ["admin", "local"]
    configure[EXCLUDE_COLLS] += ["system.profile"]

    # dump configuration for exclude option case
    if configure[INCLUDE_DBS] :
        if configure[INCLUDE_COLLS] :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[INCLUDE_DBS], configure[INCLUDE_COLLS]))
        elif configure[EXCLUDE_COLLS] :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[INCLUDE_DBS], configure[EXCLUDE_COLLS]))
        else :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[INCLUDE_DBS]))
    elif configure[EXCLUDE_DBS]:
        if configure[INCLUDE_COLLS] :
```

```
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[EXCLUDE_DBS], configure[INCLUDE_COLLS]))
        elif configure[EXCLUDE_COLLS] :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[EXCLUDE_DBS], configure[EXCLUDE_COLLS]))
        else :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[EXCLUDE_DBS]))
    else :
        if configure[INCLUDE_COLLS] :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT],
configure[INCLUDE_COLLS]))
        elif configure[EXCLUDE_COLLS] :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE],
configure[COMPARISION_COUNT],configure[EXCLUDE_COLLS]))
        else :
            log_info("Configuration [sample=%s, count=%d, include=%s,
includeColls=%s]" % (configure[SAMPLE], configure[COMPARISION_COUNT]))

    try :
        src, dst = MongoCluster(srcUrl), MongoCluster(dstUrl)
        print("[src = %s]" % srcUrl)
        print("[dst = %s]" % dstUrl)
        src.connect()
        dst.connect()
    #except (Exception, e):
    except Exception as e:
        print(e)
        log_error("create mongo connection failed %s|%s" % (srcUrl, dstUrl))
        exit()

    if check(src, dst):
        print("SUCCESS")
        exit(0)
    else:
        print("FAIL")
```

```
    exit(-1)

src.close()
dst.close()
```