마이그레이션

Tibero
대용량 이관 가이드

TMIP009

Website
www.tmaxtibero.com

안내서 정보
안내서 제목: Tibero 대용량 이관 가이드
발행일: 2024-02-14
소프트웨어 버전: Tibero7 FS02
안내서 버전: 1.0

제, 개정 이력

안내서 버전	개정일자	개정 사유 및 내용	비고
1.0	2024.02.14	최초 제정	작성자:윤준수

목록

제 1장 개요

- 본 장에서는 대용량 데이터 이관 방법 및 고려 사항에 대해서 설명한다.

1.1. 기본 개념

- 이기종 데이터베이스를 이관할 때 대용량 테이블이 있다면 이관 방법을 통해 수행 시간을 단축할 수 있다.
- 고객사의 환경(네트워크, Disk I/O, 데이터 속성)을 고려하여 적절한 방법을 선택한다.

> **다음은 이관 방법들의 특징에 대한 설명이다.**

이관 방법	특징
T-UP	• GUI 기반의 클라이언트이다. (네트워크 오버헤드가 커서 대용량 데이터를 이관할 경우 권장하지 않는다) • DPL을 지원한다.
Table Migrator	• 테이블 단위로 데이터를 옮기는데 최적화된 툴이다. • SELECT 옵션의 CONDITION, PARTITION을 지원한다. • DPL, P-DPL을 지원한다.
DB Link + DPI	• SOURCE DB에 맞는 Gateway를 설치하고 DB LINK를 설정한다. • APPEND 힌트를 통해 DPI를 지원한다.
SAM 파일 + tbLoader	• 변화가 없는 테이블에 대해 SAM 파일로 내려 tbLoader를 이용하여 데이터를 로드한다. • DPL, P-DPL을 지원한다. • External Table + DPI를 이용할 수 있다.

> **다음은 각 데이터 이관 방법들이 지원하는 방식에 대한 설명이다.**

방식	T-UP	Table Migrator	DB Link	tbLoader
Parallel Processing	O	O	O	O
Select Condition	X	O	O	X
Direct Path Load	O	O	O	O
Multi Thread	O	O	X	O
Rows Per Commit	X	X	X	O

> **대용량 데이터를 이관할 때 고려해야 할 사항은 아래와 같다.**

- 충분한 네트워크 대역을 확보한다.
- IO 대역을 고려하여 스케줄 또는 DOP를 적용한다.
- DPL, Parallel DPL을 최대한 활용한다.
- DPL, Parallel DPL이 불가능할 때에는 CPL을 병렬로 수행한다. (Multi Insert)
- Index, Constraint, Trigger 등은 데이터 이관이 끝난 다음 생성한다.

1.2. 주요 용어

- **CPL (Conventional Path Loading)**
  - 컬럼 데이터를 읽어 컬럼 배열에 담고 일괄 처리 방식으로 데이터를 로드하는 방법이다.

- **DPL (Direct Path Loading)**
  - 특정 컬럼의 데이터 타입에 맞게 데이터를 컬럼 배열 형태로 만든 다음 블록 형식기를 거쳐 로드하는 방법으로 다음과 같은 제약 사항이 있다.
    - Parallel DPL의 경우 Tibero 5 r69848 이후부터 지원한다.
    - SUBPARTITIONED TABLE은 DPL 또는 Parallel DPL을 미지원한다.
    - IOT는 DPL 또는 Parallel DPL을 미지원한다.
    - Partitioned INDEX의 경우 Normal Index만 DPL을 지원한다.
    - LONG 컬럼이 포함된 255개 이상의 Column을 가진 테이블은 DPL을 미지원한다.
    - LOB 계열의 컬럼이 있는 경우 DPL만 지원한다.
    - INDEX가 생성되어 있는 경우 DPL만 지원한다.

- **DOP (Degree Of Parallelism)**
  - 병렬도라고 하며, 단일 병렬 작업에서 사용되는 병렬 실행 서버의 수이다.

제 2장 대용량 데이터 이관 예제

- 본 장에서는 대용량 데이터 이관 방법의 예제를 설명한다.

2.1. 개요

- 데이터 이관은 다양한 방법을 이용하여 수행할 수 있고 이로 인해 수행 속도를 최적화 시킬 수 있다.
- 단, 테이블에 대한 데이터 이관만 가능하므로 Object에 대한 이관은 수동으로 수행해야 한다.

> **다음은 예제에서 사용하는 테이블 정보이다.**

```
CREATE TABLE DBTECH.LINEITEM (
L_ORDERKEY NUMBER,
L_PARTKEY NUMBER,
L_SUPPKEY NUMBER,
L_LINENUMBER NUMBER,
L_QUANTITY NUMBER(15,2),
L_EXTENDEDPRICE NUMBER(15,2),
L_DISCOUNT NUMBER(15,2),
L_TAX NUMBER(15,2),
L_RETURNFLAG CHAR(1),
L_LINESTATUS CHAR(1),
L_SHIPDATE DATE,
L_COMMITDATE DATE,
L_RECEIPTDATE DATE,
L_SHIPINSTRUCT CHAR(25),
L_SHIPMODE CHAR(10),
L_COMMENT VARCHAR(44)
) NOLOGGING;
```

2.2. Table Migrator 예제

> 다음은 Table Migrator 방법으로 대용량 데이터를 이관하는 순서이다.

1. **migrator.properties 파일에 속성값을 설정한다.**

```
$ cat migrator.properties

########################
# Source DB Connection
########################
```

```
# Source DB type (Optional, generic | default | ORACLE)
# ORACLE option needs SOURCE_LOGIN_AS property
# GENERIC option needs SOURCE_DRIVER property
SOURCE_TYPE=ORACLE

# Source DB JDBC driver class name
# Use with SOURCE_TYPE=GENERIC option
SOURCE_DRIVER=oracle.jdbc.OracleDriver
#SOURCE_DRIVER=com.mysql.jdbc.Driver

# Source DB JDBC connection URL (Optional,
Default=jdbc:oracle:thin:@localhost:1521:orcl)
SOURCE_URL=jdbc:oracle:thin:@localhost:1539:orcl
#SOURCE_URL=jdbc:mysql://127.0.0.1:3306/mysql
# Source DB login ID (Optional, Default=sys)
#SOURCE_USER="user1"
SOURCE_USER=tibero

# Source DB login password (Optional, Default=oracle)
SOURCE_PASSWORD=tmax

# Oracle source DB login option (Optional, Default=sysdba)
# Use with SOURCE_TYPE=ORACLE option
SOURCE_LOGIN_AS=NORMAL

# Source table schema (Optional)
# Default value is the same with SOURCE_USER
#SOURCE_SCHEMA=SCOTT
#SOURCE_SCHEMA="dbo"
#SOURCE_SCHEMA=

# Source table name (Mandatory)
#SOURCE_TABLE=tableName1
#SOURCE_TABLE="tableName2"
#SOURCE_TABLE=bt_scen_tran
SOURCE_TABLE=t1

# Ignore deadlock error that occur during migration for the tables created using the
dedup option
# Use when BATCH_THRESHOLD=1
# If not, Data integrity is not guaranteed
SOURCE_TABLE_DEDUPLICATE=N

########################
# Target DB Connection
########################

# Target DB type (Optional, DEFAULT)
TARGET_TYPE=DEFAULT

# Target DB JDBC driver class name
#TARGET_DRIVER=com.tmax.tibero.jdbc.TbDriver
TARGET_DRIVER=com.m.internal.jdbc.MInternalDriver
```

```
# Target DB JDBC connection URL (Optional,
Default=jdbc:internal:thin:@localhost:8629:tibero)
#TARGET_URL=jdbc:tibero:thin:@192.1.5.63:39208:tibero
TARGET_URL=jdbc:internal:thin:@localhost:59166:tibero

# Target DB login ID (Optional, Default=sys)
#TARGET_USER="user1"
TARGET_USER=tibero

# Target DB login password (Optional, Default=tibero)
TARGET_PASSWORD=tmax

# Target table schema (Optional)
# Default value is the same with TARGET_USER
# TARGET_SCHEMA=TIBERO

# Target table name (Optional)
# Default value is the same with SOURCE_TABLE
# TARGET_TABLE=t1

##################
# Data Extractor
##################
# Use parallel Extracting (Optional, Y | N Default=N)
EXTRACT_PARALLEL=N

# Number of data extractor thread (Optional, Default=1)
EXTRACT_THREAD_COUNT=1

# Use TSN(or SCN) option (Optional)
#SELECT_TSN=17322052

# Where conditions (Optional)
# ex. column1 > 10 AND column2 = 'abc'
#SELECT_CONDITION= c1>1

# Fetch size of source table cursor (Optional, Default=1024)
# Some DB is not supported this feature.
#SELECT_FETCH_SIZE=1024

# Source table partition name (Optional)
#SELECT_PARTITION=

# Source table subpartition name (Optional)
#SELECT_SUBPARTITION=subpar1
#SELECT_SUBPARTITION="subpart2"
#SELECT_SUBPARTITION=

# Ignore source/target DB character set (Optional, y | N )
SELECT_AS_BYTE=N

# db_file_multiblock_read_count (Optinal, Option for Oracle), Defalut=-1)
```

```
DB_FILE_MULTIBLOCK_READ_COUNT=-1

# Use parallel excution of SQL statements (Optinal, Option for Oracle), Defalut=N)
USE_PARALLEL_HINT=N

# Use with USE_PARALLEL_HINT=Y (Defalut=-1)
PARALLEL_HINT_CNT=-1

###############
# Data Loader
###############

# Use direct path loading (Optional, cpl | DPL)
# CPL means conventional path loading
# DPL means direct path loading
INSERT_METHOD=dpl

# Use commit size option(Optinal, Default = commit once after loading)
# Use with INSERT_METHOD=CPL option
# set commit_size-1 to commit only once
COMMIT_SIZE=-1

# Use index disable option(Optional, Default = N, Y/N)
INDEX_DISABLE=N

# Use batch insert (Optional, Y | n)
# Use with INSERT_METHOD=CPL option
INSERT_BATCH=Y

# batch threshold size (Optional, Default=32)
BATCH_THRESHOLD=128

# Use parallel loading (Optional, Y | n)
INSERT_PARALLEL=N

# Number of data loader thread (Optional, Default=4)
# Use with INSERT_PARALLEL=Y option
INSERT_THREAD_COUNT=1

# Target table partition name (Optional)
#INSERT_PARTITION=

# Specifies whether to keep the column names of the source table case-sensitive.
(Optional, Y | n )
# If set to Y, the original column name format will be preserved by enclosing it in
double quotes.
# If set to N, the column name format will follow the database naming rules.
INSERT_COLUMN_NAME_CASE_SENSITIVE=N

# Use zero length string data migration as null (Optional, Y | n, Default=Y, Option
ignored for Oracle)
# If set to N, the zero length data will be inserted as ' '.
INSERT_ZERO_LENGTH_STRING_AS_NULL=Y
```

```
# Trim leading and trailing pad character of CHAR type data. (Optional, Y | n,
Default=N)
#TRIM=

###############
# Common
###############

# Use MetaData Checking for DPL. (Optional, Default=Y)
CHECK_COLUMN_METADATA=N

# Log
LOG_LEVEL=INFO
LOG_FILE_NAME=tableMigrator.log
#LOG_DIR=
```

❗참고

- 전체 건수를 균등하게 나눌 수 있는 조건을 검토한다.(SELECT_CONDITION 부분에 설정)
- 만족하는 조건이 있는 경우 SQL문을 나눈 후에 병렬 세션(터미널 또는 다른 컴퓨터)으로 수행하며 Temp와 Undo Tablespace 사용량의 모니터링이 필요하다.

2. **Background Process를 이용하여 병렬 프로세싱으로 migrator를 수행한다.**

```
$ cat run.sh
# dbtech.lineitem 테이블 (조건 l_linestatus='O'에 해당하는 데이터)
nohup migrator.sh SOURCE_TYPE=ORACLE \
SOURCE_URL=jdbc:oracle:thin:@localhost:1521:ORCL \
SOURCE_USER=dbtech SOURCE_PASSWORD=dbtech SOURCE_LOGIN_AS=NORMAL \
SOURCE_SCHEMA=dbtech SOURCE_TABLE=lineitem \
TARGET_URL=jdbc:tibero:thin:@localhost:8629:tibero \
TARGET_USER=dbtech TARGET_PASSWORD=dbtech SELECT_CONDITION="l_linestatus='O'" >
migrator_a_set.log &

# dbtech.lineitem 테이블 (조건 l_linestatus='F'에 해당하는 데이터)
nohup migrator.sh SOURCE_TYPE=ORACLE \
SOURCE_URL=jdbc:oracle:thin:@localhost:1521:ORCL \
SOURCE_USER=dbtech SOURCE_PASSWORD=dbtech SOURCE_LOGIN_AS=NORMAL \
SOURCE_SCHEMA=dbtech SOURCE_TABLE=lineitem \
TARGET_URL=jdbc:tibero:thin:@localhost:8629:tibero \
TARGET_USER=dbtech TARGET_PASSWORD=dbtech SELECT_CONDITION="l_linestatus='F'" >
migrator_b_set.log &
```

3. **수행 결과와 로그 파일을 확인한다.**

```
# dbtech.lineitem 테이블의 조건 l_linestatus='O' migrator 수행 로그 확인
$ cat migrator_a_set.log
=== Parameters ===
```

```
PROPERTY_FILE=migrator.properties
SOURCE_TYPE=ORACLE
SOURCE_URL=jdbc:oracle:thin:@localhost:1521:ORCL
SOURCE_USER=dbtech
SOURCE_PASSWORD=************
SOURCE_LOGIN_AS=NORMAL
SOURCE_SCHEMA=dbtech
SOURCE_TABLE=lineitem
TARGET_TYPE=DEFAULT
TARGET_URL=jdbc:tibero:thin:@localhost:8629:tibero
TARGET_USER=dbtech
TARGET_PASSWORD=************
SELECT_CONDITION=l_linestatus='F'
SELECT_FETCH_SIZE=1024
SELECT_AS_BYTE=N
INSERT_METHOD=DPL
INSERT_BATCH=Y
INSERT_PARALLEL=Y
INSERT_THREAD_COUNT=4

==================

[E0]1373322363068 - STARTED
[L0]1373322363189 Loader started
[L1]1373322363192 Loader started
[L2]1373322363196 Loader started
[L3]1373322363198 Loader started
[L1] 1000
[L0] 1000
[L3] 1000
[L2] 1000
... 중략 ...
[E0]1373322793890 - TOTAL Extracted ROWS: 3004998
[L3]1373322793913 TOTAL Loaded ROWS: 617366
[L0]1373322793921 TOTAL Loaded ROWS: 893541
[L1]1373322793924 TOTAL Loaded ROWS: 898889
[L2]1373322793935 TOTAL Loaded ROWS: 595202
Loading is ended.
Elapsed Time : 431636
ADD BATCH : 0
EXECUTE BATCH : 0
[VERIFICATION] Source Info : dbtech.lineitem: 3004998 Rows, Target Info :
dbtech.lineitem : 3004998 Rows [OK]

# dbtech.lineitem 테이블의 조건 l_linestatus='F' migrator 수행 로그 확인
$ cat migrator_b_set.log
=== Parameters ===
    PROPERTY_FILE=migrator.properties
SOURCE_TYPE=ORACLE
SOURCE_URL=jdbc:oracle:thin:@localhost:1521:ORCL
SOURCE_USER=dbtech
SOURCE_PASSWORD=************
SOURCE_LOGIN_AS=NORMAL
```

```
SOURCE_SCHEMA=dbtech
SOURCE_TABLE=lineitem
TARGET_TYPE=DEFAULT
TARGET_URL=jdbc:tibero:thin:@localhost:8629:tibero
TARGET_USER=dbtech
TARGET_PASSWORD=************
SELECT_CONDITION=l_linestatus='O'
SELECT_FETCH_SIZE=1024
SELECT_AS_BYTE=N
INSERT_METHOD=DPL
INSERT_BATCH=Y
INSERT_PARALLEL=Y
INSERT_THREAD_COUNT=4

==================

[E0]1373322363042 - STARTED
[L0]1373322363168 Loader started
[L1]1373322363172 Loader started
[L2]1373322363176 Loader started
[L3]1373322363215 Loader started
[L1] 1000
[L0] 1000
[L3] 1000
[L2] 1000
...(중략)
[E0]1373322795287 - TOTAL Extracted ROWS: 2996217
[L1]1373322795290 TOTAL Loaded ROWS: 767331
[L3]1373322795303 TOTAL Loaded ROWS: 759888
[L2]1373322795312 TOTAL Loaded ROWS: 739667
[L0]1373322795314 TOTAL Loaded ROWS: 729331
Loading is ended.
Elapsed Time : 432951
ADD BATCH : 0
EXECUTE BATCH : 0
[VERIFICATION] Source Info : dbtech.lineitem: 2996217 Rows, Target Info :
dbtech.lineitem : 2996217 Rows [OK]
```

2.3. DB Link 예제

- DB Link로 insert ... select ... 구문을 수행할 때 parallel 힌트를 사용하면 select는 serial하게 동작하고, insert 일 때만 parallel로 동작한다.
- 따라서 DB Link 원본 데이터가 존재하는 서버의 사양이 좋을 경우 parallel 힌트를 사용하여 view를 생성하고 해당 view를 Tibero에서 select하는 방식으로 이관하면 성능 향상을 기대 할 수 있다.

> **다음은 DB Link 방법으로 대용량 데이터를 이관하는 순서이다. 만약, 세션이 2개일 경우 동시에 수행해야 한다.**

1. **전체 건수를 균등하게 나눌 수 있는 조건을 검토한다.**
    - 만족하는 조건이 있는 경우 SQL문을 나눈 후에 병렬 세션(터미널 또는 다른 컴퓨터)으로 수행하며 Temp와 Undo Tablespace 사용량의 모니터링이 필요하다.

2. **SQL문에 append, parallel 힌트를 사용한다.**
    - append
        - Insert하는 대상 테이블에 대해서 nologging 옵션이 적용되어 있어야 한다.

```
-- nologging 테이블 확인
select logging from user_tables where table_name = 'LINEITEM';

-- DB Link를 통해 ITAS 수행 (appned)
insert /*+ append */ into dbtech.lineitem b
select * from lineitem@olink where l_linestatus='O';
```

    - parallel
        - 힌트를 적용하기 위해 해당 세션에 parallel dml 옵션을 enable해야 한다.

```
-- dml 병렬 파라미터 활성화
alter session enable parallel dml;

-- DB Link를 통해 ITAS 수행 (parallel)
insert /*+ parallel(b 4) */ into dbtech.lineitem b
select * from lineitem@olink where l_linestatus='O';
```

❗참고

- Select하는 부분에 병렬 수행을 적용하기 위해서는 DB Link의 테이블에 대해서 parallel을 적용한 View를 만들어 조회해야 한다.

> **다음은 세션이 2개일 경우 DB Link를 수행하는 예이다.**

```
-- 세션1과 세션2에서 입력한 최종 데이터
SQL> select l_linestatus, count(*) from dbtech.lineitem@olink group by l_linestatus;

L_LINESTATUS  COUNT(*)
------------ ----------
F               2996217
O               3004998

2 rows selected
```

- **세션 1수행**

- L_LINESTATUS가 '0'에 대해서 Insert한다.

```
SQL> set timing on
SQL> alter session enable parallel dml;

Session altered.

Total elapsed time 00:00:00.005235

SQL> insert /*+ append parallel(b 4) */ into dbtech.lineitem b
select * from lineitem@olink where l_linestatus='O';

3004998 rows inserted.

Total elapsed time 00:07:07.806974

SQL> commit;

Commit completed.

Total elapsed time 00:00:00.000307
```

- **세션 2수행**
  - L_LINESTATUS가 'F'에 대해서 Insert한다.

```
SQL> set timing on
SQL> alter session enable parallel dml;

Session altered.

Total elapsed time 00:00:00.005235

SQL> insert /*+ append parallel(b 4) */ into dbtech.lineitem b
select * from lineitem@olink where l_linestatus='F';

2996217 rows inserted.

Total elapsed time 00:07:01.692934

SQL> commit;

Commit completed.

Total elapsed time 00:00:00.000307
```

- **COMMIT된 데이터 확인**

```
SQL> select count(*) from dbtech.lineitem;

COUNT(*)
----------
6001215

1 row selected.
```

2.4. tbLoader 예제

- Load할 SAM 파일을 n등분하고 직접 로딩과 병렬 프로세싱을 동시에 사용한다.
- 데이터를 로딩하는 방식으로 디스크에서 wait만 발생하지 않는다면 최적의 수행 속도가 보장된다.
- Loader 방법을 사용할 경우 다음의 사항에 주의한다.
  - Source 데이터의 unload 시간을 고려한다.
  - Source 서버에서 Tartget 서버로 파일 전송 시간을 고려한다.

> **다음은 tbLoader 방법으로 대용량 데이터를 이관하는 순서이다.**

---

1. **Unload(spool 또는 unload 유틸을 이용하여 데이터 unload) 및 SAM 파일을 전송한다.**
2. **Load할 SAM 파일을 n등분 하여 n개의 SAM 파일을 준비한다.**
3. **각 SAM 파일을 읽는 n개의 컨트롤 파일을 작성하고, 작성할 때 append 옵션을 사용한다.**
   - 다음은 2개의 컨트롤 파일을 작성하는 방법이다.

```
# SAM 파일을 특정 기준으로 N개로 파일을 나눈다.
# 나눈 파일 마다 tbLoader Control 파일을 만들어준다.

# aa 컨트롤 파일
$ cat lineitem.ctl.aa
load data
infile 'lineitem.tbl.aa'
logfile 'lineitem.tbl.aa.log'
badfile 'lineitem.tbl.aa.bad'
append
into table lineitem
fields terminated by '|'
(L_ORDERKEY,
L_PARTKEY,
L_SUPPKEY,
L_LINENUMBER,
L_QUANTITY,
L_EXTENDEDPRICE,
L_DISCOUNT,
L_TAX,
```

```
L_RETURNFLAG,
L_LINESTATUS,
L_SHIPDATE,
L_COMMITDATE,
L_RECEIPTDATE,
L_SHIPINSTRUCT,
L_SHIPMODE,
L_COMMENT)

# ab 컨트롤 파일
$ cat lineitem.ctl.ab
load data
infile 'lineitem.tbl.ab'
logfile 'lineitem.tbl.ab.log'
badfile 'lineitem.tbl.ab.bad'
append
into table lineitem
fields terminated by '|'
(L_ORDERKEY,
L_PARTKEY,
L_SUPPKEY,
L_LINENUMBER,
L_QUANTITY,
L_EXTENDEDPRICE,
L_DISCOUNT,
L_TAX,
L_RETURNFLAG,
L_LINESTATUS,
L_SHIPDATE,
L_COMMITDATE,
L_RECEIPTDATE,
L_SHIPINSTRUCT,
L_SHIPMODE,
L_COMMENT)
```

4. **Background Process를 이용하여 병렬 프로세싱으로 tbloader를 실행하고, tbloder 명령어에 'direct=y dpl_parallel=y' 옵션을 사용한다.**

```
$ cat run.sh
# aa 파일 tbLoader 명령
nohup tbloader userid=dbtech/dbtech@tibero control=lineitem.ctl.aa
direct=y dpl_parallel=y &

# ab 파일 tbLoader 명령
nohup tbloader userid=dbtech/dbtech@tibero control=lineitem.ctl.ab
direct=y dpl_parallel=y &
```

❗참고

- dpl_parallel 옵션은 하나의 테이블에 대해 병렬 프로세싱으로 tbloader를 수행할 때 사용하며 Tibero5r70450 이후 버전에서 사용이 가능하다.

> 다음은 2개의 SAM 파일 및 컨트롤 파일을 가지고 tbLoader를 수행하는 예이다.

- **세션 1 수행**

```
-- lineitem.tbl.aa.log:

tbLoader 7

TmaxTibero Corporation Copyright (c) 2020-. All rights reserved.

Data File : lineitem.tbl.aa
Bad File  : lineitem.tbl.aa.bad
Discard File : lineitem.tbl.aa.dsc

Table 'LINEITEM' was loaded from the data file.

COLUMN_NAME                     POSITION  DATATYPE
------------------------------ ---------- ----------------
L_ORDERKEY                              1  CHARACTER
L_PARTKEY                               2  CHARACTER
L_SUPPKEY                               3  CHARACTER
L_LINENUMBER                            4  CHARACTER
L_QUANTITY                              5  CHARACTER
L_EXTENDEDPRICE                         6  CHARACTER
L_DISCOUNT                              7  CHARACTER
L_TAX                                   8  CHARACTER
L_RETURNFLAG                            9  CHARACTER
L_LINESTATUS                           10  CHARACTER
L_SHIPDATE                             11  DATE
L_COMMITDATE                           12  DATE
L_RECEIPTDATE                          13  DATE
L_SHIPINSTRUCT                         14  CHARACTER
L_SHIPMODE                             15  CHARACTER
L_COMMENT                              16  CHARACTER

Table LINEITEM :
----------------
      0 Rows were skipped.
3000608 Rows were requested to load.
3000608 Rows were loaded successfully.
      0 Rows were failed to load because of some errors
      0 Rows were discarded.

0 Rows were failed to load because of some errors

Elapsed time was: 00:00:33.670898
```

- **세션 2 수행**

```
-- lineitem.tbl.ab.log:

tbLoader 7

TmaxTibero Corporation Copyright (c) 2020-. All rights reserved.

Data File : lineitem.tbl.ab
Bad File : lineitem.tbl.ab.bad
Discard File : lineitem.tbl.ab.dsc

Table 'LINEITEM' was loaded from the data file.

COLUMN_NAME                     POSITION  DATATYPE
------------------------------ ---------- ----------------
L_ORDERKEY                              1  CHARACTER
L_PARTKEY                               2  CHARACTER
L_SUPPKEY                               3  CHARACTER
L_LINENUMBER                            4  CHARACTER
L_QUANTITY                              5  CHARACTER
L_EXTENDEDPRICE                         6  CHARACTER
L_DISCOUNT                              7  CHARACTER
L_TAX                                   8  CHARACTER
L_RETURNFLAG                            9  CHARACTER
L_LINESTATUS                           10  CHARACTER
L_SHIPDATE                             11  DATE
L_COMMITDATE                           12  DATE
L_RECEIPTDATE                          13  DATE
L_SHIPINSTRUCT                         14  CHARACTER
L_SHIPMODE                             15  CHARACTER
L_COMMENT                              16  CHARACTER

Table LINEITEM :
----------------

      0 Rows were skipped.
3000607 Rows were requested to load.
3000607 Rows were loaded successfully.
      0 Rows were failed to load because of some errors
      0 Rows were discarded.


0 Rows were failed to load because of some errors

Total Elapsed Time: 00:00:33.795058
```

성능 개선 방법

> 다음은 각 상황에 따른 성능 개선 방법에 대한 설명이다.

- **Conventional Path Load 방식으로 이관할 경우**
  - Tibero 클라이언트는 데이터가 지정한 크기만큼 바인딩되기 전까지 서버로 업로드되지 않으므로 대용량의 데이터를 업로드할 때 효율적으로 사용할 수 있다.
  - tbLoader에서 제공하는 아래의 옵션을 사용하여 이관한다.

옵션	설명
rows	• 사용자가 대용량 데이터를 업로드할 때 Commit을 수행할 레코드 개수를 지정하는 파라미터이다. • 단, tbLoader는 성능을 고려하여 지정한 레코드의 개수를 정확히 맞추어서 서버로 데이터를 보내지는 않는다.
bindsize	• 배열에 사용될 최대 바이트 수를 지정하는 파라미터이다. (기본값 : 64KB)

- **인덱스가 생성된 테이블에 데이터를 Upload할 경우**
  - Direct Path Load 방식으로 데이터를 로드할 때 테이블에 존재하는 인덱스의 생성 방법을 MULTI INSERT 방식과 FAST BUILD 방식 중에 하나 선택할 수 있다.

> 다음은 인덱스 생성 방법에 대한 설명이다.

파라미터	설명
MULTI INSERT INDEXES	• 인덱스를 여러 개의 레코드 단위로 최적화 하여 한 번에 생성하는 방식이다.
FAST BUILD INDEXES	• 기존의 인덱스를 무시하고 데이터 파일의 데이터를 모두로드 하여 다시 생성하는 방식이다.

❗참고

- tbLoader 유틸리티를 사용하기 전에 대상 테이블에 기존 데이터가 많으면 MULTI INSERT 방식으로 인덱스를 생성하는 것이 유리하고 그 외는 FAST BUILD 방식이 유리하다.
- 단, tbLoader 유틸리티를 사용하여 DPL할 때 데이터 중복 등의 문제로 인해서 인덱스가 Unusable 상태로 변경될 수 있다.

> 인덱스 생성 방법을 지정하는 컨트롤 파일의 세부 내용은 다음과 같다.

```
# MULTI INSERT INDEXES 적용
```

```
$cat control.ctl
LOAD DATA
...
MULTI INSERT INDEXES
...
(...)

# FAS BUILD INDEXES 적용
$cat control.ctl
LOAD DATA
...
FAST BUILD INDEXES
...
(...)
```