TPT

실습 환경 구성

- 실습용 csv 다운로드. Firefox 브라우저를 열고 https://storage.googleapis.com/clearscape_analytics_demo_data/TPT/index_2020.csv 주소를 입력하여 다운로드. 이후 tpt_script 디렉토리로 해당 csv 이동

```
mkdir tpt_script
mv /root/download/*.csv /root/tpt_script
```

- 실습 DB 생성

```
CREATE DATABASE irs AS PERMANENT = 120e6, -- 120MB
SPOOL = 120e6;
```

- 실습 사용자 생성.

```
CREATE USER "tpt_user"
AS
PASSWORD = "tpt_user",
permanent = 100000000,
spool = 2000000000,
temporary = 500000000,
default database = "irs"
;

GRANT ALL ON irs TO tpt_user WITH GRANT OPTION;
```

TPT(Teradata Parallel Transporter) 개요

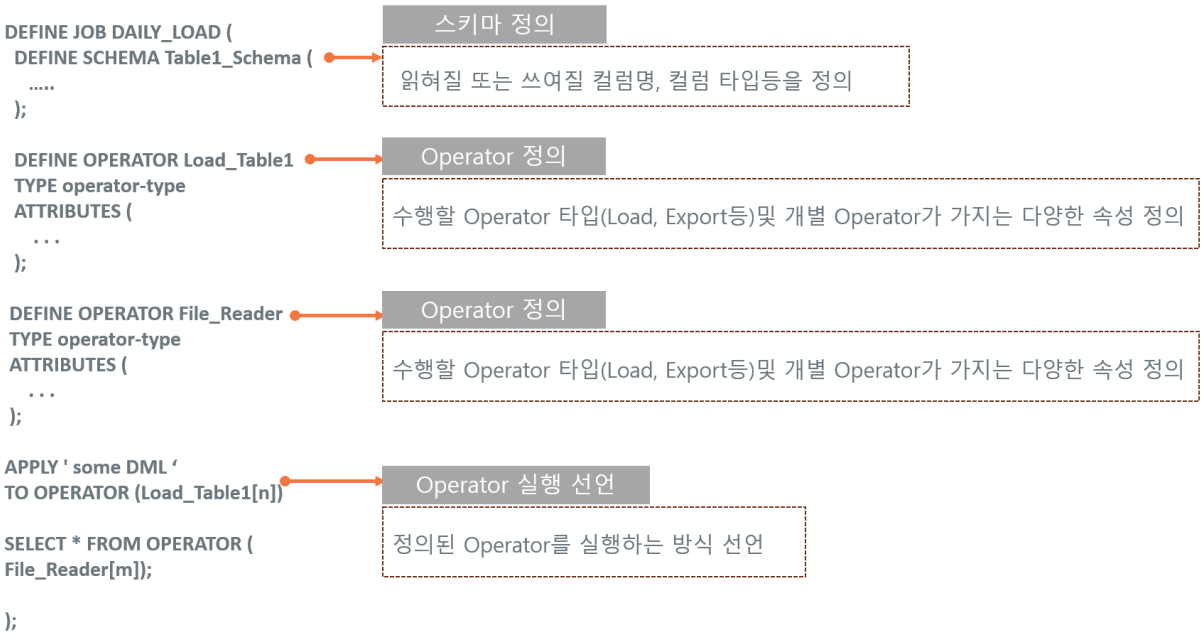
- TPT는 빠르게 대용량의 Flat file, CSV와 같은 파일들을 Teradata로 import(Load)하거나 Teradata의 테이블을 파일로 export(Unload)하거나 DB to DB 이관용도로 주로 적용됨.
- TPT는 기존 FastLoad등의 utility들을 통합되어 사용할 수 있게 재 개발 됨.
- TPT는 Script 기반 및 Library 형태의 API로 사용할 수 있음.
- TPT는 ETL Tool이 가지는 다양한 기능적인 특성 담고 있으며, 이 때문에 별도의 ETL Tool과도 유연하게 통합이 가능
- load(import), export외에 ddl 생성, upsert(update/insert), delete, stream 데이터의 upsert등 다양한 기능을 지원하며 이들은 모두 Operator로 제공됨.
- TPT는 많은 기능을 가지고 있으며 또한 개별 기능에 대한 상세한 조절이 가능함. 그리고 이를 수행하기 위해 SQL과 유사한 Script 를 통해서 내부 프로세스를 기술하고 관리함. 하지만 이 때문에 많은량의 TPT 스크립트를 작성해야 하는 단점이 있음.
- 실 적용에서는 Parameter 파일이나 Script Templates를 활용하여 단순화된 적용을 추구

TPT Script의 주요 구성 요소.

- TPT는 SQL과 유사한 형태의 Script 언어를 사용하여 extract, transformation, load등의 작업을 수행합니다. Script의 주요 구성은 Job, Schema, Operator등을 정의(Define)하고, 이들을 Apply 적용하여 복잡한 추출/변환/로딩을 수행할 수 있게 합니다.
- TPT Script는 크게 보자면 두개 섹션으로 나눌 수 있습니다. 하나는 선언 섹션(Declarative Section)으로 Define 문장으로 구성되어 있습니다. 선언 섹션은 TPT Script의 첫번째 부분으로 TPT에 사용되는 모든 Object들에 대한 정의(Define)를 담고 있습니다.

- 다른 하나는 수행 섹션(Executable Section)으로 TPT가 수행할 개별 Job을 Apply Syntax를 사용하여 정의합니다. 하나의 Job은 여러개의 Step 별로 수행할 수 있으며 개별 step들은 Apply Syntax로 상세 수행을 기술합니다.

- DEFINE JOB
  - Defines the overall job. 전체 job을 Define하며 JOB 내부에서 SCHEMA, OPERATOR를 DEFINE하고 APPLY를 기재
- DEFINE SCHEMA
  - 읽어들일/출력할 Data의 구조와 타입을 기재(Schema는 Producer Operator에서 참조)
  - 여러 schema들을 하나의 script에 기재 가능.
- DEFINE OPERATOR
  - Job에서 사용될 여러가지 유형의(Load, Export, DDL등) Operator들을 기재
  - Operator는 TPT에서 다양한 작업을 수행하는 주체임. Operator Process가 수행할 때 필요로 하는 여러 속성값을 함께 기재.
- APPLY
  - 기재된 Schema, Operator 들을 참조하여 실제 TPT의 다양한 Operation을 수행하는 문장



TPT 스크립트 만들고 수행해 보기 - DDL

- Table을 생성하는 TPT 스크립트를 생성. Operator에서는 Apply에서 사용될 메타 정보를 기술하고, Apply에서 DDL 생성.
- 아래 Script를 simple_tab_job_01.tpt 로 저장함.

```
/* DDL을 생성하는 간단한 스크립트 */
DEFINE JOB CREATE_SIMPLE_TABLE_JOB
DESCRIPTION '테이블을 생성하는 Job'
(
  /* 간단한 테스트를 위해 SCHEMA 정보는 생략
  DEFINE SCHEMA ....
  */
```

```
  /* DDL 생성을 위한 메타 정보를 기재 */
    DEFINE OPERATOR DDL_OPERATOR
    TYPE DDL
    ATTRIBUTES
    (
     VARCHAR TdpId           = '127.0.0.1' /*System Name*/
     , VARCHAR UserName        = 'tpt_user'    /*USERNAME*/
     , VARCHAR UserPassword    = 'tpt_user' /*Password*/
     , VARCHAR Errorlist             ='3807' /* skip 'Table does not exist error' */
     );

    STEP SETUP_SIMPLE_TABLE
    (
         APPLY /* APPY TO OPERATOR 기술 */
         ('DROP TABLE irs.irs_returns;'), /* 개별 SQL 문장은 ; 을 맨마지막에 기술해 줌. */
         /*
         APPLY 내에 여러개의 수행 SQL을 기술할 수 있음. 개별 SQL 문장은 APPLY내에서 ()로 기술되고
         comma(,)로 나누어지며, 맨 마지막 ()는 comma를 붙이지 않음.
         */
         ('CREATE TABLE irs.irs_returns  (
              return_id INT,
              filing_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              ein INT,
              tax_period INT,
              sub_date VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              taxpayer_name VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              return_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              dln BIGINT,
              object_id BIGINT
             )
             PRIMARY INDEX ( return_id );'
        )
        /* APPLY 는 TO OPERATOR와 같이 이어지는 구문임
        (TO OPERATOR는 Define된 Operator이며, SELECT와 같은 내장 OPERATOR도 같이 이어 질 수 있음
        세미콜론(;)은 APPLY 뒤가 아니라 연결되는 OPERATOR 마지막에 기술되어야 함.
        TO OPERATOR 내에 위에서 DEFINE한 OPERATOR를 입력.
        */
        TO OPERATOR (DDL_OPERATOR);
     ); /* END OF STEP */
); /* END OF JOB */
```

- TPT는 tbuild 명령어를 이용하여 Script 형태로 수행

```
tbuild -f simple_tab_job_01.tpt
```

- DDL_OPERATOR에서 ERRORLIST는 기술된 오류가 발생할 경우 전체 JOB을 Abort하지 않고 오류를 skip후 다음 프로세스를 적용하기 위해서 기술. 여러 OPERATOR 별로 서로 다른 Error Skip 루틴이 있으며 DDL 생성의 경우 ERRORLIST에 회피할 Error Code값을 기술함 error code 3807은 object does not exist 임.
- 만약 ERRORLIST에 회피할 오류 코드를 기술하지 않을 경우, 기존에 생성되지 않는 Table을 Drop table 시 오류가 발생하여 Create Table을 수행하지 않고 전체 Job이 종료됨.

TPT 스크립트 만들고 수행해 보기 - CSV 파일을 로드하여 DB에 입력

- 아래 SQL을 수행하여 테이블을 재 생성.

```
DROP TABLE irs.irs_returns;

CREATE MULTISET TABLE irs.irs_returns  (
              return_id INT,
              filing_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              ein INT,
              tax_period INT,
              sub_date VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              taxpayer_name VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              return_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              dln BIGINT,
              object_id BIGINT
);

SELECT COUNT(*) FROM irs.irs_returns;
```

- csv 파일을 읽어서 기존에 생성된 irs.irs_return 테이블에 입력하는 tpt 스크립트 작성. csv 파일을 읽어서 table로 로딩하기 위해서는 테이블에 대한 schema 정의, 수행 operator 정의, 그리고 실행에 대한 코드가 스크립트로 기재되어야 함. 아래 코드를 import_csv01.tpt로 저장

```
DEFINE JOB import_csv_fileload_01
DESCRIPTION 'Load a Teradata table from a file'
(
  /*
    Define the schema of the data in the csv file
  */
  DEFINE SCHEMA SCHEMA_IRS
  (
      in_return_id     VARCHAR(19),
      in_filing_type   VARCHAR(5),
      in_ein           VARCHAR(19),
      in_tax_period    VARCHAR(19),
      in_sub_date      VARCHAR(22),
      in_taxpayer_name VARCHAR(100),
      in_return_type   VARCHAR(5),
      in_dln           VARCHAR(19),
      in_object_id     VARCHAR(19)
  );

  DEFINE OPERATOR READ_CSV
    DESCRIPTION 'Operator to Read CSV File'
    TYPE DATACONNECTOR PRODUCER
    SCHEMA SCHEMA_IRS
    ATTRIBUTES
    (
      VARCHAR Filename            =   'index_2020.csv'  /*give file name with path*/
     ,VARCHAR Format               =   'Delimited'
     ,VARCHAR TextDelimiter        =   ','
     ,VARCHAR AcceptExcessColumns  =   'N'
     ,Integer SkipRows=1  /*skips the header in csv file*/
    );
```

```
    DEFINE OPERATOR LOAD_CSV    /*Load information*/
    DESCRIPTION 'Operator to Load CSV Data'
    TYPE LOAD
    SCHEMA *
    ATTRIBUTES
    (
      INTEGER BUFFERSIZE      = 1024,
      VARCHAR TargetTable     = 'irs.irs_returns',   /*Define target table name */
      VARCHAR TdpId           = '127.0.0.1',  /*System Name*/
      VARCHAR UserName        = 'tpt_user',     /*Username*/
      VARCHAR UserPassword    = 'tpt_user'              /*Password*/
    );

    STEP LOAD_TABLE
    (
   APPLY ('INSERT INTO irs.irs_returns (
          return_id,
          filing_type,
          ein,
          tax_period,
          sub_date,
          taxpayer_name,
          return_type,
          dln,
          object_id
      ) VALUES (
          :in_return_id,
          :in_filing_type,
          :in_ein,
          :in_tax_period,
          :in_sub_date,
          :in_taxpayer_name,
          :in_return_type,
          :in_dln,
          :in_object_id
      );')  /*Inserts records from CSV file into Target Table*/
        TO OPERATOR (LOAD_CSV)
        SELECT * FROM  operator(READ_CSV);
    ); /* END OF STEP LOAD_TABLE */
); /* END OF JOB */
```

- tbuild 명령어를 이용하여 Script 수행

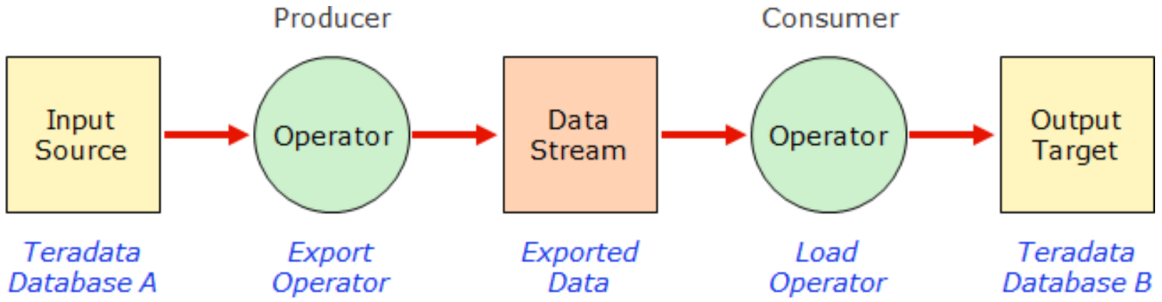
```
tbuild -f import_csv01.tpt
```

- LOAD Operation은 테이블에 레코드가 없었을 때만 수행 가능함. import_csv01.tpt를 다시 수행하여 데이터가 로딩 되는지 확인. 아래 오류 발생.

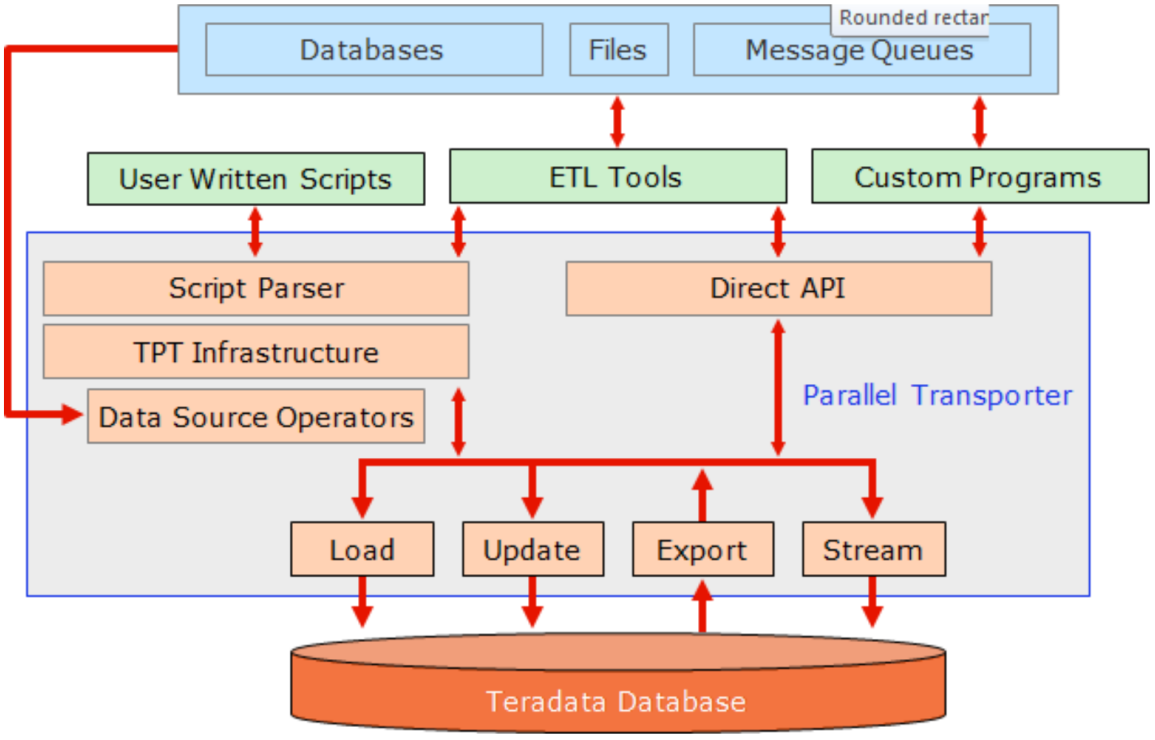
```
LOAD_CSV: TPT10508: RDBMS error 2636: irs_returns must be empty for Fast Loading
```

TPT 아키텍처 개요 및 주요 구성 요소

- TPT는 Operator와 Data Stream의 연속된 Pipeline 형태로 동작. Data stream과 Operator를 연속적으로 연결하여 Pipeline을 구성하며 이를 Parallel로 처리하여 빠른 수행 속도 보장.



- TPT는 스크립트나 Direct API와 통합된 ETL Tool을 이용하여 동작할 수 있으며, 다양한 유형의 Operator들을 통해서 데이터 이관, Load, Export, 가공, 변환 등을 작업을 빠르게 수행할 수 있음.



Operator Type(유형)

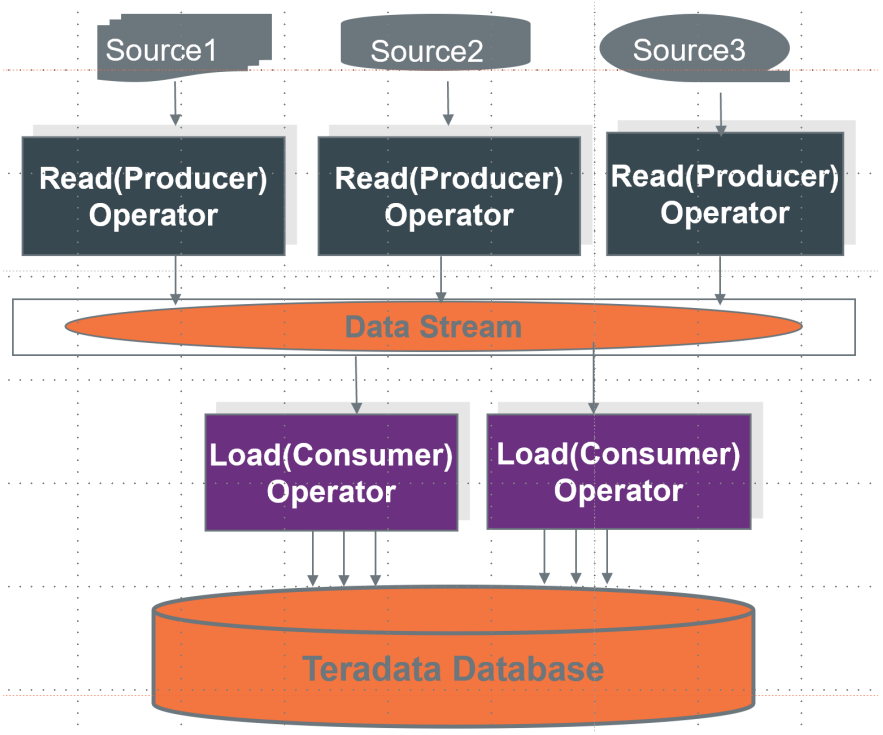
- operator는 TPT 수행 시에 로드되는 다양한 유형의 수행 라이브러리를 지칭. Operator는 TPT가 수행하는 주요 프로세스에 따라 여러 유형이 있지만, 주로 Producer와 Consumer 유형의 Operator가 대부분.
- Producer와 Consumer는 Producer→Consumer 형태로 Pipeline으로 구성되며, 단독으로 사용되는 경우는 거의 없음.
- Producer Operator
  - Producer Operator는 주로 Data Stream에 Write를 수행함.
  - Teradata DB나 외부 소스(타 DB, Flat file, CSV 등)로 부터 데이터를 읽어서 Data Stream에 기록
  - 다른 Operator가 데이터를 활용할 수 있도록 제공하는 역할을 수행.
- Consumer Operator
  - Data Stream에서 읽어 들이는 역할을 수행.
  - Teradata Table이나 외부 소스(타 DB, Flat file, CSV등)로 데이터를 Load하는 역할을 수행.

    - 다른 Operator로부터 데이터를 받아들임.
- Standalone Operator
    - Data Stream이 없으며 주로 DDL, OS Command 수행 시 사용됨.
    - 다른 Input Source들과 연관되지 않으며 독립적으로 사용.

Operator Type에 따른 세부 Operator들

- Operator 유형 Producer, Consumer, Standalone 에 따른 상세 Operator들이 있으며, Operator Type에 맞지 않은 Operator를 script에 기술 시 오류 발생.

Producer	Consumer	Standalone
Export Selector ODBC **DataConnector**	**Load** Update Inserter Stream **DataConnector**	 **DDL** Load Update OS Command



- Input Source와 Output Target은 File(Flat file, Delimeter file), RDBMS(Teradata 및 타 DB), Queue 및 다양한 데이터 Source/Target 가능

주요 세부 Operator

Operator	과거 Utility	Type Operator	개요
Load	Fastload	Consumer	빈 테이블에 대량 데이터를 Load
Update	Multiload	Consumer	대량으로 Upsert 및 Delete 수행.
Stream	TPump	Consumer	지속적인 Stream으로 Teradata Table에 SQL 프로토콜로 데이터 Load
SQL Inserter	BTEQ Import	Consumer	SQL 프로토콜로 비교적 소량의 데이터를 Table로 Load
Export	FastExport	Producer	Teradata에서 대량의 데이터를 Export

Operator	과거 Utility	Type Operator	개요
SQL Selector	BTEQ Export	Producer	SQL 프로토콜로 비교적 소량의 데이터를 Table에서 Export
ODBC Operator	OLE DB Access Module	Producer	ODBC driver를 적용하여 Data Export
DataConnector	I/Os in the SA utilities	Producer/ Consumer	Reads/writes flat files

TPT Script의 주요 구성 요소 살펴 보기

- TPT Script는 Define Schema절, Define Operator절 그리고 Apply절로 크게 구성되어 있음.

Define Schema 절

- CSV와 같은 파일, RDBMS와 같은 소스에서 데이터를 읽어 들여서 Teradata에 Load하려면 해당 데이터 소스의 Schema를 먼저 정의해 줘야 함.
- Schema는 일반적으로 Load 되어야 할 Table의 컬럼명과 컬럼타입을 아래와 같이 지정해줌

```
DEFINE SCHEMA my_table_schema
(
    ID   INTEGER,
    Name VARCHAR(50),
    Age  INTEGER
);
```

- Schema가 정의되면 Insert, Select, Update등의 Operator가 해당 Schema를 참조하여 소스 데이터를 읽고 타겟에 Load 할 수 있음.
- CSV 파일을 DB로 로딩할 경우에도 CSV 컬럼들의 컬럼명과 컬럼타입을 파악한 뒤, 타겟 Table과 매핑하여 Schema로 설정해줌. 일반적으로 타겟 Table의 컬럼명과 컬럼 타입으로 Schema 설정함.
- CSV 파일의 컬럼 타입이 명확하지 않을 경우 VARCHAR로 선언해도 무방. 데이터 LOAD시 Table 컬럼 타입으로 변환되어 입력됨. 하지만 선언된 VARCHAR의 길이는 실제 CSV 컬럼 길이보다 작아서는 안됨.

Define Operator 절

- TPT에서 사용할 Operator의 타입 및 Operator 동작을 위한 attribute를 설정
- Operator Type은 적용 업무에 따라 설정해 주며, Operator Type에 따라 동작을 위한 다양한 attribute 속성도 달라짐.
- 예를 들어 CSV 파일을 읽어 들여서, DB 테이블에 로딩 하기 위해서는 CSV 파일을 읽어 들여서 Data Stream에 전달해주는 Operator와 Data Stream을 읽어서 DB Table에 Load해주는 두 개의 Operator가 필요.
- CSV 파일을 읽어 들이는 Operator Type은 DATACONNECTOR PRODUCER 로 설정하며, 읽어 들일 데이터의 포맷 정보를 가지는 Schema명, 그리고 DATACONNECTOR PRODUCER의 동작을 위해서 필요한 ATTRIBUTE들을 기술함(파일명, DELIMTER값, Header값 제외 등)
- ATTRIBUTE를 기술할 때 마치 정적 타입 언어의 변수 선언과 같이 속성 타입을 varchar/integer와 같은 형태로 기재해 줌에 유의

```
DEFINE OPERATOR READ_CSV
    DESCRIPTION 'Operator to Read CSV File'
    TYPE DATACONNECTOR PRODUCER
    SCHEMA my_table_schema
    ATTRIBUTES
    (
      VARCHAR Filename            =   'index_2020.csv'  /*give file name with path*/
     ,VARCHAR Format               =   'Delimited'
```

```
     ,VARCHAR TextDelimiter        =   ','
     ,VARCHAR AcceptExcessColumns  =   'N'
     ,Integer SkipRows=1  /*skips the header in csv file*/
    );
```

- Data Stream을 읽어서 DB에 Load하는 역할을 하는 Operator Type은 주로 LOAD를 사용. LOAD는 대용량 데이터를 빠르게 INSERT하는데 적합하며, DB 테이블의 데이터가 비어 있어야 함.
- LOAD의 경우 SCHEMA를 * 로 정의함. 이를 Deferred Schema라 하며, Producer에서 적용된 Output 데이터의 Schema를 그대로 적용하겠다는 의미임. Attribute는 DB 테이블에 입력하기 위해 LOAD Operator가 사용해야 할 주요 속성값을 정의 (타겟 테이블명, 타겟 DB 주소, 접속사용자, 패스워드등)

```
DEFINE OPERATOR LOAD_CSV    /*Load information*/
    DESCRIPTION 'Operator to Load CSV Data'
    TYPE LOAD
    SCHEMA *
    ATTRIBUTES
    (
      INTEGER BUFFERSIZE      = 1024,
      VARCHAR TargetTable     = 'irs.irs_returns',   /*Define target table name */
      VARCHAR TdpId           = '127.0.0.1',  /*System Name*/
      VARCHAR UserName        = 'tpt_user',     /*Username*/
      VARCHAR UserPassword    = 'tpt_user'              /*Password*/
    );
```

APPLY 절 (Step 절이 상위에 올 수 있음)

- APPLY절은 정의된 Operator들을 Pipeline형태로 연결하여 수행할 수 있도록 함. Operator Type에 따라 수행 스크립트가 달라짐.
- APPLY절은 OPERATOR TYPE과 수행 Pipeline에 따라서 다양한 형태로 기재 될 수 있음. 예를 들어 PRODUCER → LOAD 등을 이용해 CSV 파일을 읽고 이를 Teradata에 입력할 경우는 APPLY 절에 INSERT SELECT와 같은 형태로 수행 프로세스를 기재함.
- APPLY 절은 Operator 단일 형태로 수행될 경우는 APPLY (SQL이 필요할 경우 SQL문) TO OPERATOR(위에서 선언한 Operator명) 과 같이 기재할 수 있음. APPLY 내부의 SQL은 ( )를 이용하여 수행할 여러 개의 SQL을 정의할 수 있음.

```
DEFINE OPERATOR DDL_OPERATOR
TYPE DDL
ATTRIBUTES
(
 VARCHAR TdpId           = '127.0.0.1' /*System Name*/
 , VARCHAR UserName        = 'tpt_user'    /*USERNAME*/
 , VARCHAR UserPassword    = 'tpt_user' /*Password*/
 , VARCHAR Errorlist             ='3807' /*skip 'Table does not exist error' */
 );

APPLY /* APPY TO OPERATOR 기술 */
         ('DROP TABLE irs.irs_returns;'), /* 개별 SQL 문장은 ; 을 맨마지막에 기술해 줌. */
         /*
         APPLY 내에 여러개의 수행 SQL을 기술할 수 있음. 개별 SQL 문장은 APPLY내에서 ()로 기술되고
         comma(,)로 나누어지며, 맨 마지막 ()는 comma를 붙이지 않음.
         */
         ('CREATE TABLE irs.irs_returns  (
              return_id INT,
```

```
              filing_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              ein INT,
              tax_period INT,
              sub_date VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              taxpayer_name VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              return_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              dln BIGINT,
              object_id BIGINT
            )
            PRIMARY INDEX ( return_id );'
        )
        /* APPLY 는 TO OPERATOR와 같이 이어지는 구문임
        (TO OPERATOR는 Define된 Operator이며, SELECT와 같은 내장 OPERATOR도 같이 이어 질 수 있음
        세미콜론(;)은 APPLY 뒤가 아니라 연결되는 OPERATOR 마지막에 기술되어야 함.
        TO OPERATOR 내에 위에서 DEFINE한 OPERATOR를 입력.
        */
        TO OPERATOR (DDL_OPERATOR);
```

- Producer에서 전달 받아 Consumer로 입력하는 Pipeline 형태로 수행 시 에는 APPLY(SQL) TO Operator(위에서 선언한 Consumer Operator명) SELECT * FROM Operator(위에서 선언한 Producer Operator)와 같은 형태로 구성하여 수행할 수 있음.

```
APPLY ('INSERT INTO irs.irs_returns (
          return_id,
          filing_type,
          ein,
          tax_period,
          sub_date,
          taxpayer_name,
          return_type,
          dln,
          object_id
      ) VALUES (
          :in_return_id,
          :in_filing_type,
          :in_ein,
          :in_tax_period,
          :in_sub_date,
          :in_taxpayer_name,
          :in_return_type,
          :in_dln,
          :in_object_id
      );')  /*Inserts records from CSV file into Target Table*/
        TO OPERATOR (LOAD_CSV) /* Operator type LOAD로 선언된 Operator */
        SELECT * FROM  operator(READ_CSV); /* Operator type DATACONNECTOR PRODUCER */
```

- 여러 개의 Apply절을 순차적으로 수행하고자 할 경우에는 Step 절을 이용하여 개별 Apply 절에 대한 정의를 별도로 하는 것이 script의 직관적인 이해를 높임.

```
STEP SETUP_SIMPLE_TABLE
    (
         APPLY
         (DDL 수행 SQL)
        TO OPERATOR (DDL_OPERATOR);
     ); /* END OF STEP SETUP_SIMPLE_TABLE */
```

```
    STEP LOAD_TABLE
    (
   APPLY (INSERT 수행 SQL)  /*Inserts records from CSV file into Target Table*/
        TO OPERATOR (LOAD_CSV)
        SELECT * FROM  operator(READ_CSV);
    );
```

여러 Apply들을 순차 적용하기

- DDL → Read → Load 순으로 Proccess를 적용해 보기
- 먼저 irs.irs_return 테이블을 Drop 하여 초기화.
- 아래 스크립트를 import_csv_02.tpt로 저장.

```
DEFINE JOB import_csv_fileload_02
DESCRIPTION 'Load a Teradata table from a file'
(
  /*
    Define the schema of the data in the csv file
  */
  DEFINE SCHEMA SCHEMA_IRS
  (
      in_return_id     VARCHAR(19),
      in_filing_type   VARCHAR(5),
      in_ein           VARCHAR(19),
      in_tax_period    VARCHAR(19),
      in_sub_date      VARCHAR(22),
      in_taxpayer_name VARCHAR(100),
      in_return_type   VARCHAR(5),
      in_dln           VARCHAR(19),
      in_object_id     VARCHAR(19)
  );

  /* DDL 생성을 위한 메타 정보를 기재 */
    DEFINE OPERATOR DDL_OPERATOR
    TYPE DDL
    ATTRIBUTES
    (
       VARCHAR TdpId           = '127.0.0.1' /*System Name*/
     , VARCHAR UserName        = 'tpt_user'    /*USERNAME*/
     , VARCHAR UserPassword    = 'tpt_user' /*Password*/
     , VARCHAR Errorlist             ='3807' /*This is added to skip 'Table does not exi
     );

  DEFINE OPERATOR READ_CSV
    DESCRIPTION 'Operator to Read CSV File'
    TYPE DATACONNECTOR PRODUCER
    SCHEMA SCHEMA_IRS
    ATTRIBUTES
    (
      VARCHAR Filename            =   'index_2020.csv'  /*give file name with path*/
     ,VARCHAR Format               =   'Delimited'
     ,VARCHAR TextDelimiter        =   ','
```

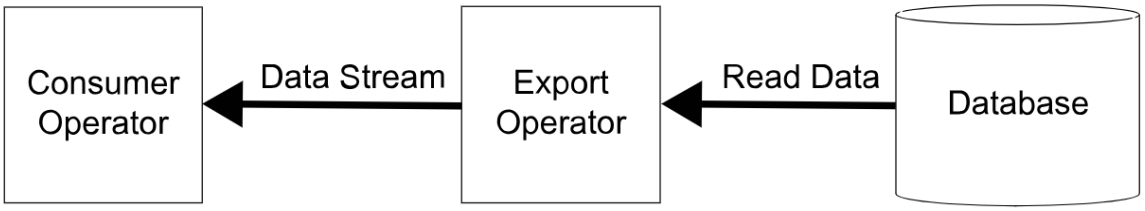
```
 ,VARCHAR AcceptExcessColumns  =   'N'
 ,Integer SkipRows=1  /*skips the header in csv file*/
);

DEFINE OPERATOR LOAD_CSV    /*Load information*/
DESCRIPTION 'Operator to Load CSV Data'
TYPE LOAD
SCHEMA *
ATTRIBUTES
(
  INTEGER BUFFERSIZE      = 1024,
  VARCHAR TargetTable     = 'irs.irs_returns',   /*Define target table name */
  VARCHAR TdpId           = '127.0.0.1',  /*System Name*/
  VARCHAR UserName        = 'tpt_user',     /*Username*/
  VARCHAR UserPassword    = 'tpt_user'              /*Password*/
);

STEP SETUP_SIMPLE_TABLE
(
     APPLY
     ('DROP TABLE irs.irs_returns;'),
     ('CREATE TABLE irs.irs_returns  (
          return_id INT,
          filing_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
          ein INT,
          tax_period INT,
          sub_date VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
          taxpayer_name VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
          return_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
          dln BIGINT,
          object_id BIGINT
         )
         PRIMARY INDEX ( return_id );'
    )
    TO OPERATOR (DDL_OPERATOR);
 ); /* END OF STEP SETUP_SIMPLE_TABLE */

STEP LOAD_TABLE
(
APPLY ('INSERT INTO irs.irs_returns (
       return_id,
       filing_type,
       ein,
       tax_period,
       sub_date,
       taxpayer_name,
       return_type,
       dln,
       object_id
  ) VALUES (
      :in_return_id,
      :in_filing_type,
      :in_ein,
      :in_tax_period,
      :in_sub_date,
```

```
          :in_taxpayer_name,
          :in_return_type,
          :in_dln,
          :in_object_id
      );')  /*Inserts records from CSV file into Target Table*/
        TO OPERATOR (LOAD_CSV)
        SELECT * FROM  operator(READ_CSV);
    ); /* END OF STEP LOAD_TABLE */
); /* END OF JOB */
```

- tbuild로 수행.

```
tbuild -f import_csv02.tpt
```

Table의 데이터를 Export하여 csv 파일로 만들기



- Schema, Export Operator, Consumer Operator를 이용하여 Table의 데이터를 읽어서 CSV 파일로 내려 받기
- Schema는 읽어들일 Table의 컬럼명/타입을 정의
- Table의 데이터를 읽어서 Data Stream에 기록하는 Operator는 Export 를 사용. Export Operator는 대용량 Table의 데이터를 빠르게 읽어서 Data Stream에 기록함. Export Operator는 Consumer가 아니라 Producer임에 유의
- Export Operator는 DB와 Interface를 하므로 위에서 정의한 SCHEMA 정보도 SCHEMA 속성으로 매핑 시키며 접속 DB정보등의 Attribute들도 매핑 필요. 또한 SelectStmt 속성으로 SELECT 문을 지정하여 Export 할 테이블/컬럼명/where 조건들을 기술할 수 있음.
- Data Stream을 읽어서 File로 기록하는 Operator는 DATACONNECTOR CONSUMER를 이용.
- DATACONNECTOR CONSUMER는 Data Stream을 읽어서 File에 기록하는 역할을 수행. 이를 위해 Attribute로 write되는 파일명, Delimeter, 파일모드등을 설정해줌.
- export_csv01.tpt 파일을 만들고 아래 내용을 저장.

```
DEFINE JOB export_csv_unload_01
DESCRIPTION 'Export a Teradata table from a file'
(
  /*
    Define the schema of irs.irs_returns table
  */
  DEFINE SCHEMA SCHEMA_IRS
  (
      return_id     INT,
      filing_type   VARCHAR(5),
      ein           INT,
      tax_period    INT,
      sub_date      VARCHAR(100),
      taxpayer_name VARCHAR(100),
      return_type   VARCHAR(5),
```

```
      dln           BIGINT,
      object_id     BIGINT
  );

  /* DB 테이블을 읽어오는 EXPORT OPERATOR 정의 */
    DEFINE OPERATOR EXPORT_OPERATOR
  TYPE EXPORT
  SCHEMA SCHEMA_IRS
  ATTRIBUTES
  (
    VARCHAR TdpId           = '127.0.0.1',  /*System Name*/
      VARCHAR UserName        = 'tpt_user',     /*Username*/
      VARCHAR UserPassword    = 'tpt_user',             /*Password*/
    VARCHAR SelectStmt = 'SELECT *  FROM irs.irs_returns;'
  );

  DEFINE OPERATOR FILE_WRITER
  TYPE DATACONNECTOR CONSUMER
  SCHEMA *
  ATTRIBUTES
  (
      VARCHAR DirectoryPath = '/root/tpt_script',
      VARCHAR FileName = 'irs_returns_export.csv',
      VARCHAR Format = 'DELIMITED',
    VARCHAR TextDelimiter = ',',
    VARCHAR OpenMode = 'Write'
    );

    APPLY TO OPERATOR (FILE_WRITER) /* 파일 write시 SQL이 필요하지 않음 */
    SELECT * FROM OPERATOR(EXPORT_OPERATOR); /* Export Operator 호출 */

);
```

- tbuild로 수행.

```
tbuild -f export_csv01.tpt
```

TPT 스크립트에 Parameter 적용하기 - 주요 장점

- 스크립트의 재 사용성 및 유연성: 여러 다른 상황에서도 파라미터만 동적으로 변경하여 동일한 TPT 스크립트를 재활용.
- 보다 쉬운 스크립트 작성: Parameter를 별도 관리하여 서로 다른 dataset나 table, 기타 환경들을 쉽게 customization 할 수 있음.
- 보안: 민감한 정보를 hard-coding이 아닌 별도의 파일로 관리.

Job내에 SET 키워드로 Parameter 설정하기

- . SET Parameter명 = 값; 으로 설정하고 해당 파라미터는 @파라미터명 으로 참조함.
- 아래 스크립트를 create_ddl_02.tpt로 생성 후 실행.

```
/* DDL을 생성하는 간단한 스크립트 */
DEFINE JOB CREATE_SIMPLE_TABLE_JOB_02
DESCRIPTION '테이블을 생성하는 Job'
```

```
(
    /* JOB내에 Parameter를 SET으로 설정 */
  SET TgtTableName='irs.irs_returns';

  /* DDL 생성을 위한 메타 정보를 기재 */
    DEFINE OPERATOR DDL_OPERATOR
    TYPE DDL
    ATTRIBUTES
    (
     VARCHAR TdpId           = '127.0.0.1' /*System Name*/
     , VARCHAR UserName        = 'tpt_user'    /*USERNAME*/
     , VARCHAR UserPassword    = 'tpt_user' /*Password*/
     , VARCHAR Errorlist             ='3807' /*skip 'Table does not exist error'*/
     );

    STEP SETUP_SIMPLE_TABLE
    (
         APPLY /* APPY TO OPERATOR 기술 */
         ('DROP TABLE ' || @TgtTableName ||';'), /* 개별 SQL 문장은 ; 을 맨마지막에 기술해 줌.
         /*
         APPLY 내에 여러개의 수행 SQL을 기술할 수 있음. 개별 SQL 문장은 APPLY내에서 ()로 기술되고
         comma(,)로 나누어지며, 맨 마지막 ()는 comma를 붙이지 않음.
         */
         ('CREATE MULTISET TABLE '|| @TgtTableName ||'  (
              return_id INT,
              filing_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              ein INT,
              tax_period INT,
              sub_date VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              taxpayer_name VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              return_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              dln BIGINT,
              object_id BIGINT
             )
             PRIMARY INDEX ( return_id );'
        )
        /* APPLY 는 TO OPERATOR와 같이 이어지는 구문임
        (TO OPERATOR는 Define된 Operator이며, SELECT와 같은 내장 OPERATOR도 같이 이어 질 수 있음
        세미콜론(;)은 APPLY 뒤가 아니라 연결되는 OPERATOR 마지막에 기술되어야 함.
        TO OPERATOR 내에 위에서 DEFINE한 OPERATOR를 입력.
        */
        TO OPERATOR (DDL_OPERATOR);
     ); /* END OF STEP */
); /* END OF JOB */
```

별도의 파라미터 파일로 파라미터들을 지정하기

- 별도의 파라미터 파일을 만들고 tbuild 수행 시 -f 옵션으로 tpt 스크립트를, -v 옵션으로 파라미터 파일을 지정하여 수행.
- 파라미터 파일내의 파라미터를 Job Script에서 사용하려면 @Parameter 와 같은 형태로 적용해야 함.
- 아래 코드를 import_csv03.par 파일로 저장. parameter 파일은 ; 가 아닌 , 로 개별 파라미터들을 분리하여 저장.

```
LoadTargetTable = 'irs.irs_returns'
, ReadFileName = 'index_2020.csv'
```

```
, TdpId = '127.0.0.1'
, UserName = 'tpt_user'
, UserPassword = 'tpt_user'
```

- 아래의 Job script를 import_csv03.tpt로 저장.

```
DEFINE JOB import_csv_fileload_03
DESCRIPTION 'Load a Teradata table from a file'
(
  /*
    Define the schema of the data in the csv file
  */
  DEFINE SCHEMA SCHEMA_IRS
  (
      in_return_id     VARCHAR(19),
      in_filing_type   VARCHAR(5),
      in_ein           VARCHAR(19),
      in_tax_period    VARCHAR(19),
      in_sub_date      VARCHAR(22),
      in_taxpayer_name VARCHAR(100),
      in_return_type   VARCHAR(5),
      in_dln           VARCHAR(19),
      in_object_id     VARCHAR(19)
  );

  DEFINE OPERATOR DDL_OPERATOR
    TYPE DDL
    ATTRIBUTES
    (
     VARCHAR TdpId           = @TdpId, /*System Name*/
     VARCHAR UserName        = @UserName,    /*USERNAME*/
     VARCHAR UserPassword    = @UserPassword, /*Password*/
     VARCHAR Errorlist       ='3807' /*skip 'Table does not exist error'*/
     );

  DEFINE OPERATOR READ_CSV
    DESCRIPTION 'Operator to Read CSV File'
    TYPE DATACONNECTOR PRODUCER
    SCHEMA SCHEMA_IRS
    ATTRIBUTES
    (
      VARCHAR Filename            =   @ReadFileName  /*give file name with path*/
     ,VARCHAR Format               =   'Delimited'
     ,VARCHAR TextDelimiter        =   ','
     ,VARCHAR AcceptExcessColumns  =   'N'
     ,Integer SkipRows=1  /*skips the header in csv file*/
    );

    DEFINE OPERATOR LOAD_CSV    /*Load information*/
    DESCRIPTION 'Operator to Load CSV Data'
    TYPE LOAD
    SCHEMA *
    ATTRIBUTES
    (
      INTEGER BUFFERSIZE      = 1024,
```

```
      VARCHAR TargetTable     = @LoadTargetTable,   /*Define target table name*/
      VARCHAR TdpId           = @TdpId,  /*System Name*/
      VARCHAR UserName        = @UserName,     /*Username*/
      VARCHAR UserPassword    = @UserPassword             /*Password*/
    );

    STEP SETUP_SIMPLE_TABLE
    (
         APPLY
         ('DROP TABLE '||@LoadTargetTable ||';'),
         ('CREATE TABLE '||@LoadTargetTable ||'  (
              return_id INT,
              filing_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              ein INT,
              tax_period INT,
              sub_date VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              taxpayer_name VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC,
              return_type VARCHAR(5) CHARACTER SET LATIN NOT CASESPECIFIC,
              dln BIGINT,
              object_id BIGINT
            )
            PRIMARY INDEX ( return_id );'
        )
        TO OPERATOR (DDL_OPERATOR);
     ); /* END OF STEP SETUP_SIMPLE_TABLE */

    STEP LOAD_TABLE
    (
   APPLY ('INSERT INTO ' ||@LoadTargetTable ||' (
          return_id,
          filing_type,
          ein,
          tax_period,
          sub_date,
          taxpayer_name,
          return_type,
          dln,
          object_id
      ) VALUES (
          :in_return_id,
          :in_filing_type,
          :in_ein,
          :in_tax_period,
          :in_sub_date,
          :in_taxpayer_name,
          :in_return_type,
          :in_dln,
          :in_object_id
      );')  /*Inserts records from CSV file into Target Table*/
        TO OPERATOR (LOAD_CSV)
        SELECT * FROM  operator(READ_CSV);
    ); /* END OF STEP LOAD_TABLE */
); /* END OF JOB */
```

- tbuild를 이용하여 TPT 수행 시 parameter 파일을 -v 옵션으로 수행.

```
tbuild -f import_csv03.tpt -v import_csv03.par
```

Templates 기반 TPT 개요

- 주요 Operator들에 대한 사전에 정의된 스크립트를 Templates 형태로 제공
- Template 이름은 $Operator명 과 같은 형태로 제공
- $DDL, $LOAD, $EXPORT, $INSERTER, $SELECTOR, $FILE_READER, $FILE_WRITER, $DATACONNECTOR_PRODUCER, $DATACONNECTOR_CONSUMER, $OS_COMMAND
- 개별 Template에 대한 정의는 $tpt_install_dir(기본 /opt/teradata/client/17.20/tbuild)/templates 에 파일로 기록됨. cat \$LOAD.txt 으로 내용 확인($로 파일명이 시작하므로 \ 로 특수문자 파일명 회피 처리 필요)

```
DEFINE OPERATOR $LOAD
 DESCRIPTION 'Teradata Parallel Transporter Load Operator'
 TYPE LOAD
 SCHEMA *
 ATTRIBUTES
 (
   VARCHAR TdpId             = @LoadTdpId,
   VARCHAR UserName          = @LoadUserName,
   VARCHAR UserPassword      = @LoadUserPassword,
   VARCHAR TargetTable       = @LoadTargetTable,
   INTEGER MinSessions       = @LoadMinSessions,
   INTEGER MaxSessions       = @LoadMaxSessions,
   INTEGER BufferSize        = @LoadBufferSize,
   ...........
   VARCHAR ConnectString          = @LoadConnectString
 );
```

- OPERATOR는 Templates를 활용.
- SCHEMA는 입력 Table 스키마 또는 출력할 SelectStmt 속성 등으로 추론하거나 SELECT FROM TABLE 문으로 명시적으로 Define.
- APPLY는 job에서 정의함. Template로 정의된 Operator들의 Attribute들은 별도의 Parameter 파일로 만들거나 Parameter 변수값을 할당하여 TPT를 수행
- APPLY절에 사용되는 $INSERT 키워드는 추론을 통해 SQL Insert 문을 자동 생성.

Templates을 이용하여 CSV 파일을 Table로 로딩

- 아래 수행하여 Table 초기화

```
delete irs.irs_returns all;
```

- 아래를 import_csv04.tpt 로 저장.

```
DEFINE JOB import_csv_fileload_04
DESCRIPTION 'Load a Teradata table from a file'
(
  /* 원래 Load는 deferred schema 이지만
  여기서는 Load의 consumer가 table을 기반으로 schema를 infer */
  APPLY $INSERT TO OPERATOR($LOAD)
```

```
  SELECT * FROM OPERATOR($FILE_READER);
);
```

- 파라미터 파일에는 Templates(예를 들어 $LOAD.txt)에서 @파라미터명으로 지정된 파라미터명에 대해서 값을 부여 함. 즉 @LoadTdpId가 $LOAD.txt에 파라미터 형태로 지정되어 있으면 LoadTdpId = '127.0.0.1' 와 같이 파라미터 파일에 파라미터명 = 값을 지정.
- import_csv04.par 파일에 아래 설정 입력.

```
/*******************************************************/
/* TPT LOAD Operator setting                           */
/*******************************************************/
 LoadTdpId                  = '127.0.0.1'
,LoadUserName               = 'tpt_user'
,LoadUserPassword           = 'tpt_user'
,LoadTargetTable = 'irs.irs_returns'


/*******************************************************/
/* TPT DataConnector Producer Operator setting         */
/*******************************************************/
,FileReaderDirectoryPath    = '/root/tpt_script'
,FileReaderFileName         = 'index_2020.csv'
,FileReaderFormat           = 'Delimited'
,FileReaderOpenmode         = 'Read'
,FileReaderTextDelimiter    = ','
,FileReaderSkipRows = 1
```

- tbuild로 수행

```
tbuild -f import_csv04.tpt -v import_csv04.par
```

Templates을 이용하여 DDL Operator Pipeline과 CSV 파일 Load Pipeline을 순차적으로 수행.

- 데이터를 모두 삭제할 테이블을 파라미터명으로 입력 받아서 DDL Operator 수행. Job Script에서 파라미터를 사용하려면 @파라미터명 과 같은 형태로 Script에 기재해야 함. 아래 예에서는 파라미터 파일에 LoadTargetTable = 'irs.irs_returns' 로 지정된 파라미터를 Job Script에서 적용하기 위해 @LoadTargetTable 로 지정함.
- 아래를 import_csv05.tpt로 저장함.

```
DEFINE JOB import_csv_fileload_05
DESCRIPTION 'Load a Teradata table from a file'
(
  /*
    Define the schema of the data in the csv file
  */
  STEP SETUP_TABLE
  (
      APPLY
      ('DELETE '||@LoadTargetTable ||' ALL;')
      TO OPERATOR($DDL);
  );

  STEP LOAD_TABLE
  (
```

```
      APPLY $INSERT TO OPERATOR($LOAD) /* Load의 consumer가 schema를 infer */
      SELECT * FROM OPERATOR($FILE_READER);
     );
);
```

- 아래를 import_csv05.par로 저장함.

```
/*********************************************************/
/* TPT LOAD Operator setting                             */
/*********************************************************/
 LoadTdpId                  = '127.0.0.1'
,LoadUserName               = 'tpt_user'
,LoadUserPassword           = 'tpt_user'
,LoadTargetTable = 'irs.irs_returns'

/*********************************************************/
/* TPT DDL Operator setting                              */
/*********************************************************/
 DDLTdpId                  = '127.0.0.1'
,DDLUserName               = 'tpt_user'
,DDLUserPassword           = 'tpt_user'
,DDLErrorList = '3807'

/*********************************************************/
/* TPT DataConnector Producer Operator setting           */
/*********************************************************/
,FileReaderDirectoryPath    = '/root/tpt_script'
,FileReaderFileName         = 'index_2020.csv'
,FileReaderFormat           = 'Delimited'
,FileReaderOpenmode         = 'Read'
,FileReaderTextDelimiter    = ','
,FileReaderSkipRows = 1
```

- tbuild로 수행

```
tbuild -f import_csv05.tpt -v import_csv05.par
```