Load and Export Utilities

In this lecture

- Overview of load and export utilities.
- Comparison

FastLoad

- FastLoad utility is used to load data into empty tables.
- Since it does not use transient journals, data can be loaded quickly.
- It doesn't load duplicate rows even if the target table is a `MULTISET` table.

  Limitations
- The target table should not have secondary index, join index and foreign key reference.

Phase 1

- The Parsing engines read the records from the input file and sends a block to each AMP.
- Each AMP stores the blocks of records.
- Then AMPs hash each record and redistribute them to the correct AMP.
- At the end of Phase 1, each AMP has its rows but they are not in row hash sequence.

Phase 2

- Phase 2 starts when FastLoad receives the END LOADING statement.
- Each AMP sorts the records on row hash and writes them to the disk.
- Locks on the target table is released and the error tables are dropped.

Example

Suppose we have an `employee.txt` file with the followng records:

```
101,Mike,James,1980-01-05,2010-03-01,1
102,Robert,Williams,1983-03-05,2010-09-01,1
103,Peter,Paul,1983-04-01,2009-02-12,2
104,Alex,Stuart,1984-11-06,2014-01-01,2
105,Robert,James,1984-12-01,2015-03-09,3
```

Example (cont.)

```
LOGON 192.168.1.102/dbc,dbc;
DATABASE tduser;
BEGIN LOADING tduser.Employee_Stg
ERRORFILES Employee_ET, Employee_UV
CHECKPOINT 10;
SET RECORD VARTEXT ",";
DEFINE in_EmployeeNo (VARCHAR(10)) ,
in_FirstName (VARCHAR(30)) ,
in_LastName (VARCHAR(30)) ,
in_BirthDate (VARCHAR(10)) ,
in_JoinedDate (VARCHAR(10)) ,
in_DepartmentNo (VARCHAR(02)) ,
FILE=employee.txt;
INSERT INTO Employee_Stg
(EmployeeNo,FirstName,LastName,BirthDate,JoinedDate,
DepartmentNo)
VALUES ( :in_EmployeeNo , :in_FirstName, in_LastName ,
:in_BirthDate (FORMAT 'YYYY-MM-DD'),
:in_JoinedDate (FORMAT 'YYYY-MM-DD'), :in_DepartmentNo);
END LOADING;
LOGOFF;
```

```
FastLoad < EmployeeLoad.fl;
```

FastLoad Terms

- `ERRORFILES:` Identifies the 2 error tables that needs to be created/updated.
- `CHECKPOINT:` Defines when to take checkpoint.
- `SET RECORD:` Specifies if the input file format is formatted, binary, text or unformatted.
- `DEFINE:` Defines the input file layout.
- `FILE:` Specifies the input file name and path.
- `INSERT:` Inserts the records from the input file into the target table.
- `END LOADING:` Initiates phase 2 of the FastLoad. Distributes the records into the target table.

MultiLoad

- `MultiLoad` can load multiple tables at a time and it can also perform different types of tasks such as `INSERT`, `DELETE`, `UPDATE` and `UPSERT`.
- It can load up to 5 tables at a time and perform up to 20 DML operations in a script. The target table is not required for `MultiLoad.`
- MultiLoad supports two modes:
  - `IMPORT`
  - `DELETE`

MultiLoad: Requirements

- **Log Table:** Used to maintain the checkpoints taken during load which will be used for restart.
- **Error Tables:** These tables are inserted during load when an error occurs. First error table stores conversion errors whereas second error table stores duplicate records.
- **Log Table:** Maintains the results from each phase of MultiLoad for restart purpose.
- **Work table:** MultiLoad script creates one work table per target table. Work table is used to keep DML tasks and the input data.

MultiLoad: Phases

MultiLoad import has five phases:

- **Phase 1:** Preliminary Phase – Performs basic setup activities.
- **Phase 2:** DML Transaction Phase –Verifies the syntax of DML statements and brings them to Teradata system.
- **Phase 3:** Acquisition Phase – Brings the input data into work tables and locks the table.
- **Phase 4:** Application Phase – Applies all DML operations.
- **Phase 5:** Cleanup Phase – Releases the table lock.

Example

```
.LOGTABLE tduser.Employee_log;
.LOGON 192.168.1.102/dbc,dbc;
.BEGIN MLOAD TABLES Employee_Stg;
.LAYOUT Employee;
.FIELD in_EmployeeNo * VARCHAR(10);
.FIELD in_FirstName * VARCHAR(30);
.FIELD in_LastName * VARCHAR(30);
.FIELD in_BirthDate * VARCHAR(10);
.FIELD in_JoinedDate * VARCHAR(10);
.FIELD in_DepartmentNo * VARCHAR(02);
.DML LABEL EmpLabel;
INSERT INTO Employee_Stg
(EmployeeNo,FirstName,LastName,BirthDate
,JoinedDate,DepartmentNo)
VALUES
(:in_EmployeeNo,:in_FirstName,:in_Lastname
,:in_BirthDate, :in_JoinedDate,:in_DepartmentNo);
.IMPORT INFILE employee.txt
FORMAT VARTEXT ','
LAYOUT Employee
APPLY EmpLabel;
. END MLOAD;
LOGOFF;
```

FastLoad vs MultiLoad

FastLoad	MultiLoad
Target table should be empty	Target table need not be empty
Only one table can be loaded using a single script	Can load/update up to 5 tables
Supports only CREATE/INSERT statement	Supports up to 20 DML statements in single script
Does not support tables with SI	Supports tables with NUSI

FastExport

- `FastExport` utility is used to export data from Teradata tables into flat files.
- It can also generate the data in report format.
- Data can be extracted from one or more tables using joins.
- Since FastExport exports the data in 64K blocks, it is useful for extracting large volumes of data.

Example

```
.LOGTABLE tduser.employee_log;
.LOGON 192.168.1.102/dbc,dbc;
DATABASE tduser;
.BEGIN EXPORT SESSIONS 2;
.EXPORT OUTFILE employeedata.txt
MODE RECORD FORMAT TEXT;
SELECT CAST(EmployeeNo AS CHAR(10)) ,
CAST(FirstName AS CHAR(15)) ,
CAST(LastName AS CHAR(15)) ,
CAST(BirthDate AS CHAR(10))
FROM
Employee;
.END EXPORT;
.LOGOFF;
```

BTEQ

- `BTEQ` utility is a powerful utility in Teradata that can be used in both batch and interactive mode.
- It can be used to run any DDL statement, DML statement, create Macros and stored procedures.
- `BTEQ` can be used to import data into Teradata tables from flat file and it can also be used to extract data from tables into files or reports.

BTEQ: Commands

- `LOGON` : Used to log into Teradata system.
- `ACTIVITYCOUNT` : Returns the number of rows affected by the previous query.
- `ERRORCODE` : Returns the status code of the previous query.
- `DATABASE` : Sets the default database.
- `LABEL` : Assigns a label to a set of SQL commands.
- `RUN FILE` : Executes the query contained in a file.
- `GOTO` : Transfers control to a label.
- `LOGOFF` : Logs off from database and terminates all sessions.
- `IMPORT` : Specifies the input file path.
- `EXPORT` : Specifies the output file path and initiates the export.

Example

```
.LOGON 192.168.1.102/dbc,dbc;
DATABASE tduser;
CREATE TABLE employee_bkup
(
EmployeeNo INTEGER,
FirstName CHAR(30),
LastName CHAR(30),
DepartmentNo SMALLINT,
NetPay INTEGER
)
Unique Primary Index(EmployeeNo);
.IF ERRORCODE <> 0 THEN .EXIT ERRORCODE;
```

Example (cont.)

```
SELECT * FROM
Employee
Sample 1;
.IF ACTIVITYCOUNT <> 0 THEN .GOTO InsertEmployee;
DROP TABLE employee_bkup;
.IF ERRORCODE <> 0 THEN .EXIT ERRORCODE;
.LABEL InsertEmployee
INSERT INTO employee_bkup
SELECT a.EmployeeNo,
a.FirstName,
a.LastName,
a.DepartmentNo,
b.NetPay
FROM
Employee a INNER JOIN Salary b
ON (a.EmployeeNo=b.EmployeeNo);
.IF ERRORCODE <> 0 THEN .EXIT ERRORCODE;
.LOGOFF;
```

Example (cont.)

The above script performs the following tasks.

- Logs into Teradata System.
- Sets the Default Database.
- Creates a table called employee_bkup.
- Selects one record from Employee table to check if the table has any records.
- Drops employee_bkup table, if the table is empty.

Example (cont.)

- Transfers the control to a Label InsertEmployee which inserts records into employee_bkup table.
- Checks `ERRORCODE` to make sure that the statement is successful, following each SQL statement.
- `ACTIVITYCOUNT` returns number of records selected/impacted by the previous SQL query.