Teradata Bacis

Lesson 05: Teradata Training – FastLoad

Module Object

- Introduction about Teradata Utility
- Introduction to Fast Load
- Supporting Environment
- Key requirements for Fast Load
- Basic steps for Fast Load
- Loading Phase
- Simple Fast Load Script
- Fast Load Command

Module Object

- BEGIN LOADING Statement
- END LOADING Statement
- INSERT Statement
- Data Type Conversion in Fast Load
- Fast Load Restartibilty

Introduction about Teradata Utility

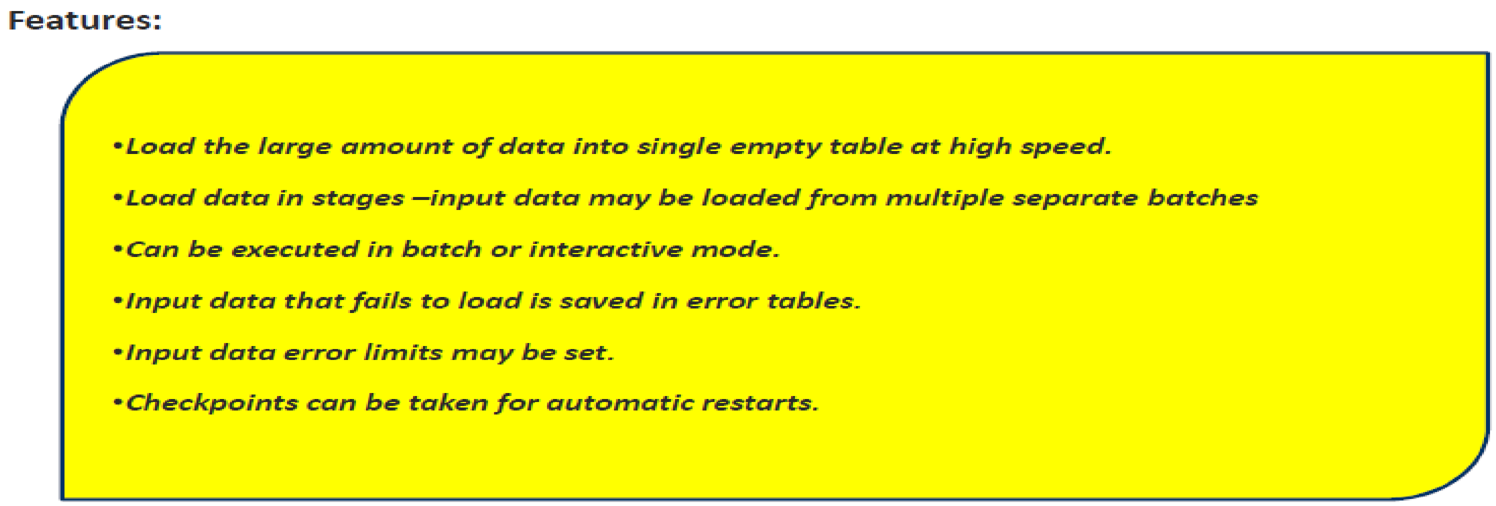
- What is the need of Teradata utilities in Data ware house.
  - Quick access to data for more timely decision making.
  - Solutions for the entire spectrum of load requirements from batch to near real time.
  - Unmatched scalability for large volume loads.
  - Fail-proof loads with checkpoint restart capabilities.
  - Proven technology from the data warehouse technology leader.
  - Integration with industry-leading ETL and ELT tools.

Introduction about Teradata Utility

- Teradata Utilities :
  - BTEQ: Help for Report formatting, Ad hoc query tool, Database administration, Best for small data volumes.
  - Multi Load :High-performance data unload in client format.
  - Fast Load: High-performance initial table load.
  - Multi Load: High-performance maintenance operations applies updates to multiple tables in single pass.
  - Apart from these teradata having other utilities like Teradata Parallel Transporter, Tpump e.t.c.

Introduction to Fast Load

- Why the Name “Fast” Load:
- Fast Load is known for its high speed in loading the large amounts of data from files to empty
- Teradata tables .This speed is achieved because it does not use the Transient Journal. Apart from these there are some other logical reasons behind this ,which makes it fast, basically it was developed to load millions of rows into a table.


Features:

•Load the large amount of data into single empty table at high speed.

•Load data in stages –input data may be loaded from multiple separate batches

•Can be executed in batch or interactive mode.

•Input data that fails to load is saved in error tables.

•Input data error limits may be set.

•Checkpoints can be taken for automatic restarts.

Introduction to Fast Load

• The Target table must initia.lly empty.

• The target table can not have Secondary Indexes(USI/NUSI),joirindexes, or Has Indexes

• Referential Integ·rity constraints are not supported.

• The target tab.le can not have Enabled Triggers.

-Duplicate rows are not loaded into target tpble(even if the table is MULTISET).

• if an AMP goes down,Fastload can not be restarted until the AMP is back online.

Supporting Environment

The Fast Load utility is supported either on either the mainframe or on network attached system(LAN)

The LAN environment supports the following Operating Systems:

- UNIX MP-RAS
- Windows 2000
- Windows 95
- Windows NT
- UNIX HP-UX
- AIX
- Solaris SPARC
- Solaris Intel

The Mainframe (Channel Attached) environment supports the following Operating Systems:

- MVS
- VM

Fast Load perform so well when it is loading millions or even billions of rows in any environment. The reason behind is

- Fast Load assembles data into 64K blocks (64,000 bytes) to load it and can use multiple sessions simultaneously.
- Fast Load taking the advantage of parallel processing.

Key requirements for Fast Load

- Fast Load can be run from either MVS/ Channel (mainframe) or Network (LAN) host. In either case, Fast Load requires three key components. case, Fast Load requires three key.
- Log Table:
  - Fast Load needs a place to record information on its progress during a load. It uses the table called Fast log in the SYSADMIN.
  - database. This table contains one row for every Fast Load running on the system. In order for your FastLoad to use this table,
  - you need INSERT, UPDATE and DELETE privileges on that table.

- Empty target table: We have already mentioned the absolute need for the target table to be empty.

Key requirements for Fast Load

- Two Error Table
  - Each Fast Load requires two error tables. These are error tables that will only be populated should errors occur during the load process. These are required by the Fast Load utility, which will automatically create.
  - The first error table is for any translation errors or constraint violations.
  - (For example, a row with a column containing a wrong data type would be reported to the first error table.)
  - The second error table is for errors caused by duplicate values for Unique Primary Indexes (UPI).
  - Fast Load will load just one occurrence for every UPI. The other occurrences will be stored in this table. However, if the entire row is a duplicate, Fast Load counts it but does not store the row. These tables may be analyzed later for troubleshooting should errors
  - occur during the load. For specifics on how you can troubleshoot, see the section below titled, "What Happens When Fast Load Finishes."

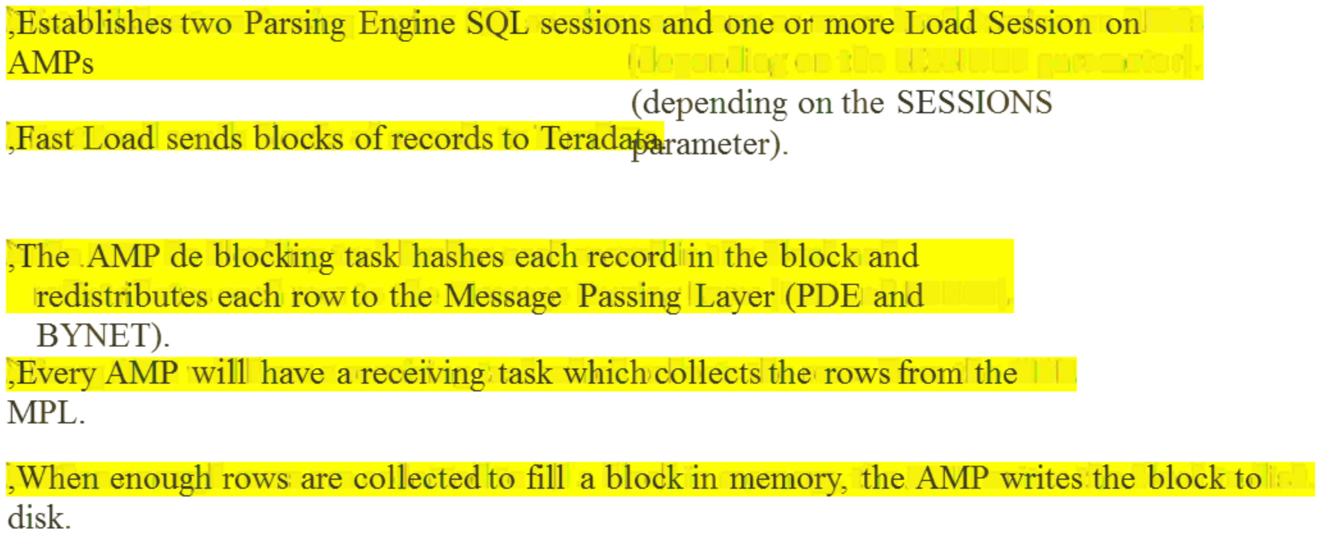
Basic steps for Fast Load

- Before run the Fast load   Script , below points should be keep in mind.
  - Logging onto Teradata
  - Defining the Teradata table that you want to load (target table)
  - Defining the INPUT data file
  - Telling the system to Start loading
  - Telling the system to End loading

- As we defined in kev requirement section Empty Teradata table is required to load the data.
- Required the source information file, \lvhere the data is going to load.
  » Acq uisirUo rn Pha s.e (:Phase ·1)
  » Ap pl]j catti o n/'E nd L.o·ad ilng/S·ort Phase(P has e2 J

Acquisition Phase

- For each Fast. load job, there are two SQl. sessions, one for handling SQL requests and the other for log handling. table re start-related operations. There are also load sessions, established for each Fast load job that can be specified in a Fast. Load script via the SESSIONS command.
  - Steps foUows, i1n Phaser



,Establishes two Parsing Engine SQL sessions and one or more Load Session on AMPs (depending on the SESSIONS parameter).

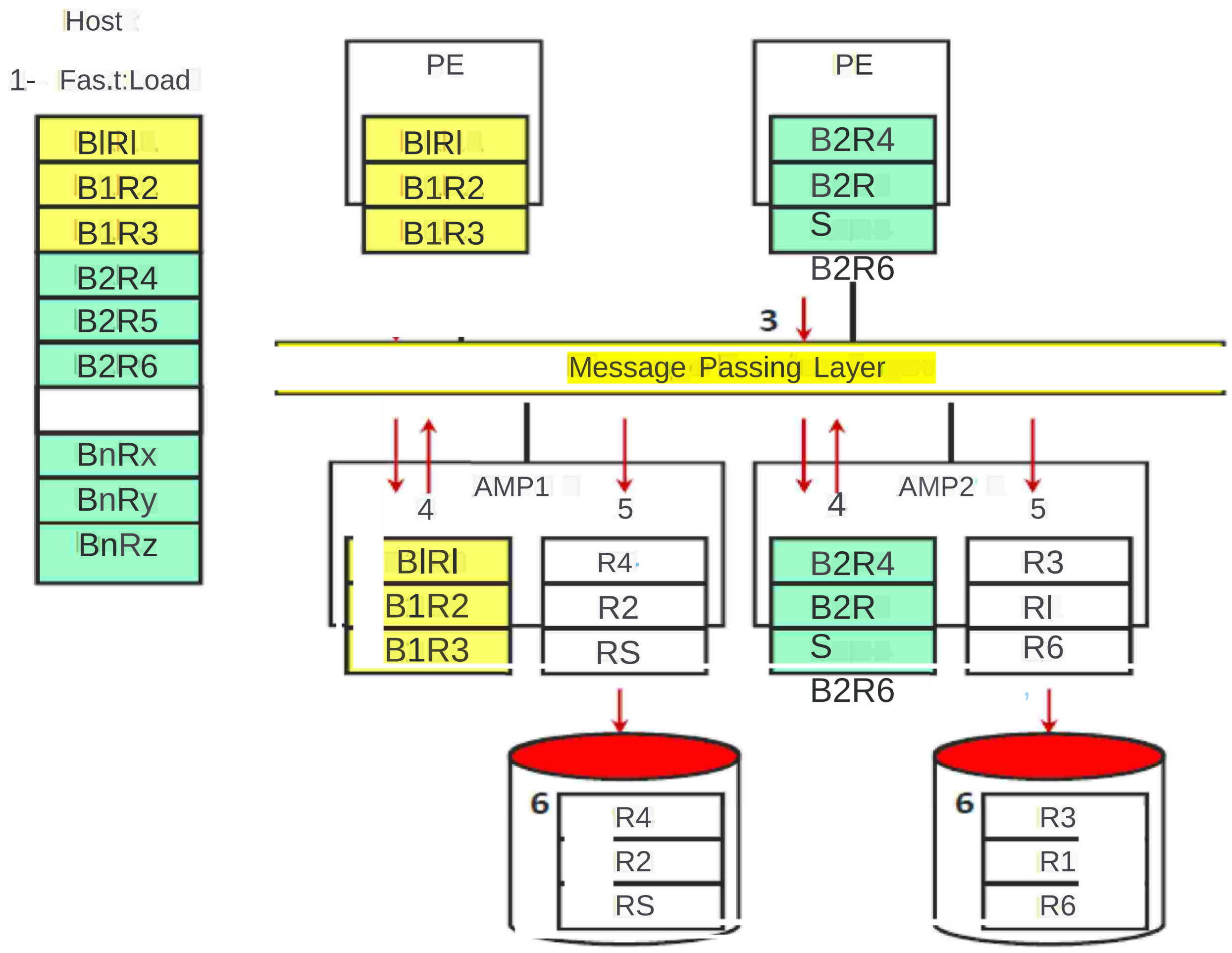
,Fast Load sends blocks of records to Teradata

,The AMP de blocking task hashes each record in the block and redistributes each row to the Message Passing Layer (PDE and BYNET).

,Every AMP will have a receiving task which collects the rows from the MPL.

,When enough rows are collected to fill a block in memory, the AMP writes the block to disk.

Fast Load Phase 1

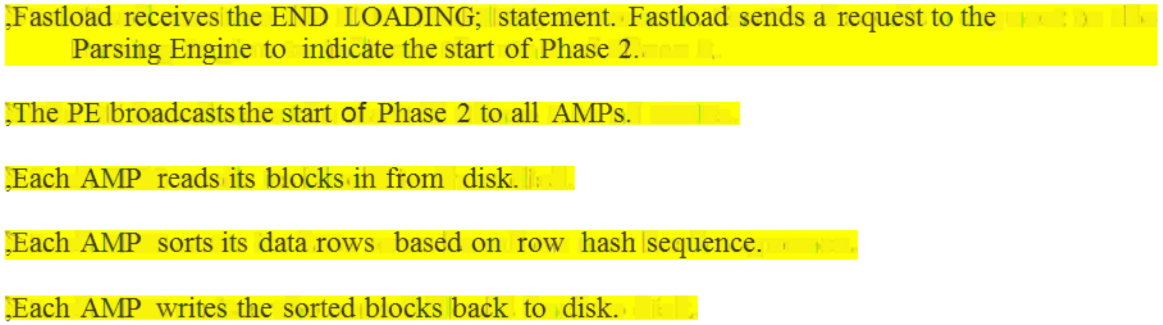


Phase 1

- Fasttoad uses one SQIL session to define A.MP steps.
- The PE sends a block to each AMP which stories blocks of unsorted data records,
- AMPs hash each record and redistribute them to the AMP responsible for the hash value.
- At the end of Phase 1, each AMP has the rows it: should hav1e, but thie rows. are not in row hash sequence.

Loading Phase

- The second phase of fastload has each AMP [in parallel] reading the data blocks from disk, sorting the data
- rows based on row hash, and writing the blocks back out to PERM space.



Fastload receives the END LOADING; statement. Fastload sends a request to the Parsing Engine to indicate the start of Phase 2.

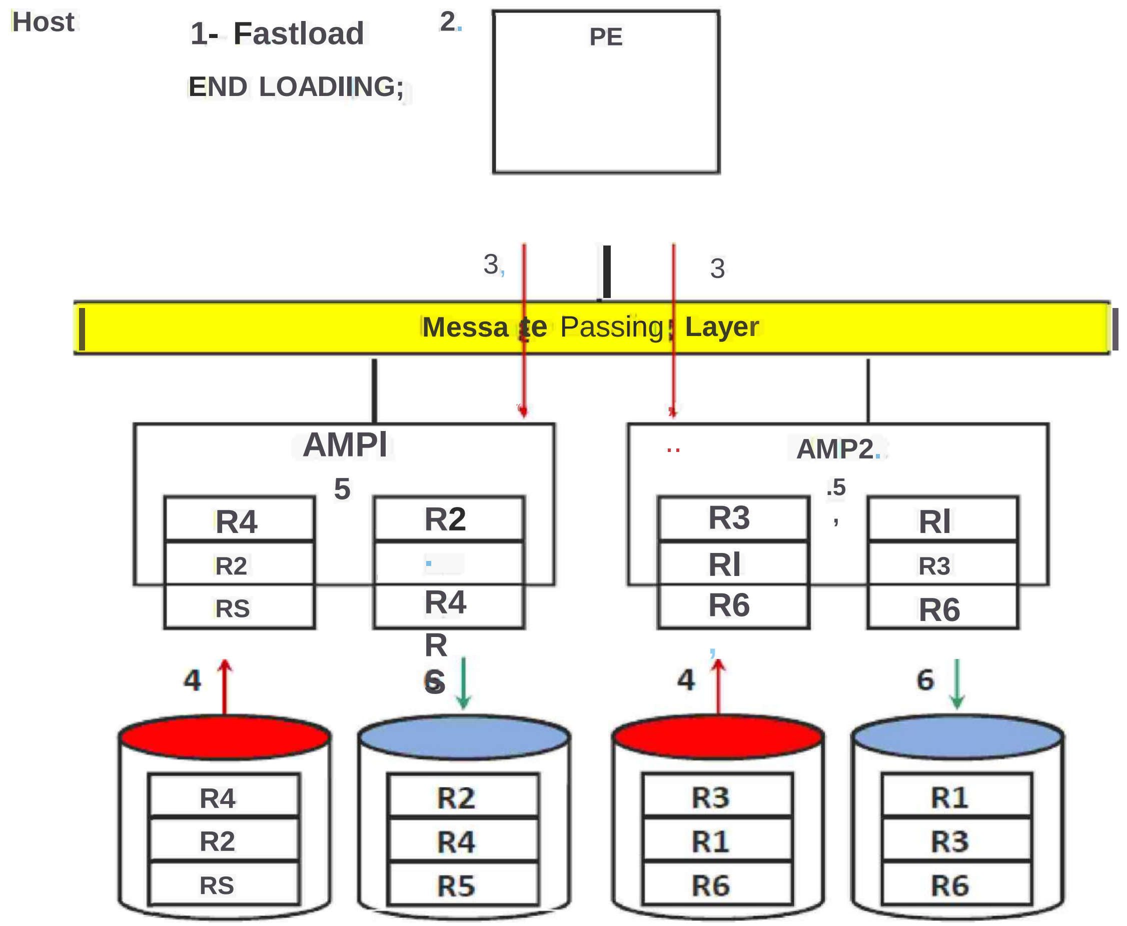
The PE broadcasts the start of Phase 2 to all AMPs.

Each AMP reads its blocks in from disk.

Each AMP sorts its data rows based on row hash sequence.

Each AMP writes the sorted blocks back to disk.

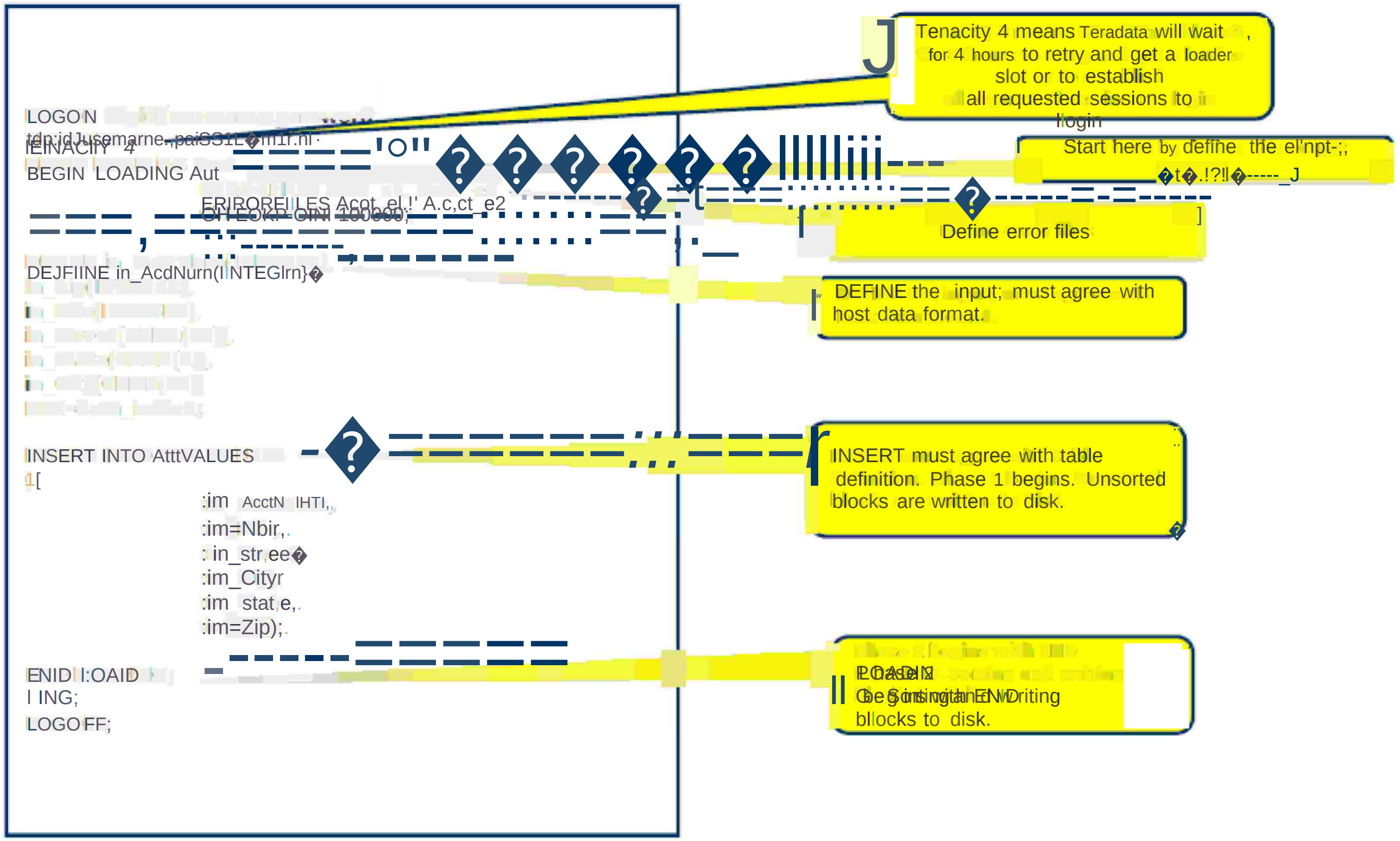
Fast Load Phase 2



Phase 2

- When tllhe Fastload job receives END LOADING; statement Fastload starts Phase 2.
- Ea,cllh AMP sorts the target table!' puts the rows into blocks!' and wri1tes the blocks to disk.
- **Fall** back rnws are then generated i1f required.
- Table data is avai lable when Phase 2 completes,

Simple Fast Load Script



```
LOGON tdp;id/username,paSS1L�m1r.nl
TENACITY 4
BEGIN LOADING Aut
    ERRORFILES Acct_el.!' A.c,ct_e2
    CHECKPOINT 100000;
DEFINE in_AcdNurn(INTEGlrn}�
in
in
in
in
in

INSERT INTO AtttVALUES
1[
            :im  AcctN  IHTI,
            :im=Nbir,
            : in_str ee�
            :im_Cityr
            :im  stat e,
            :im=Zip);

END LOADING;
LOGOFF;
```

Tenacity 4 means Teradata will wait , for 4 hours to retry and get a loader slot or to establish all requested sessions to login

Start here by define the el'npt-;, �t�.!?!l�-----_J

Define error files

DEFINE the input; must agree with host data format.

INSERT must agree with table definition. Phase 1 begins. Unsorted blocks are written to disk.

Phase 2 begins with END LOADING. Sorting and writing blocks to disk.

BEGIN Loading Systems

BEGIN LOADING [db name.] table name	**Name of Table**
ERROR FILES [db name.] Err_Table_l; [db name.] Err_Table_2;	**Name of the two error tables**
[CHECKPOINT integer]	**Optional Check point interval**

User should have privileges i1r1 order to execute the fast load.

For Target Table: select and insertltreate, Drop, 01r Delete i1f required)

For Error Table: CRIEATE JAHLE

END LOADING;

PI Table

,Indicates that all data rows have been transmitted.

,Begins Phase 2 or Sort Phase processing.

,Omission implies:

- The load is incomplete and will be restarted later.
- This causes the table that is being loaded to become "Fastload paused,"
- If you attempt to access a table (via SQL) that is in a "Fastload paused"state, you will get the error.

NoP'I Table

,There is no Sort in Phase 2 for NoPI table. Phase 2 is very fast for a NoPI table.

,While a NoPI target table is being loaded, users can view the table with an ACCESS Lock.This is possible because rows are always appended at the end of a NoPI table.

,The following SELECT will succeed for NoPI table.

```
LOCKING ROW FOR ACCESS SELECT * FROM
Orders_nopi;
```

INSERT Statement

```
DEFlIN.IE Accouint:_Num beri(:INTEG
ER),
                 Nt11mbe1f'l{IINITEG
                 ER),
                 Sit1reet:(CHAR1(25)),
                 Ci't.y(CHAR(25)),
                 Sitate1(CHAR(:2.)),
HLE = data_file3;Zip_Code(I
11INISERT INITO Accounits
                 NTEGIER)
(
                 Aocountt:_Number,
                 Nlt11mbe
                 ir,
                 Sit1reet:,
                 Ci't.V11
                  State,
                 Zip_Code
)
VALUES.
(:Accounntt:_INumber,
:Number,
:Street,
:Cit�
:State,
:Zip_Gode);
```

The "wildcard'forrnat may be used to construct the names in the INSERT statement. The field names are constructed from the D[)/D table definition.

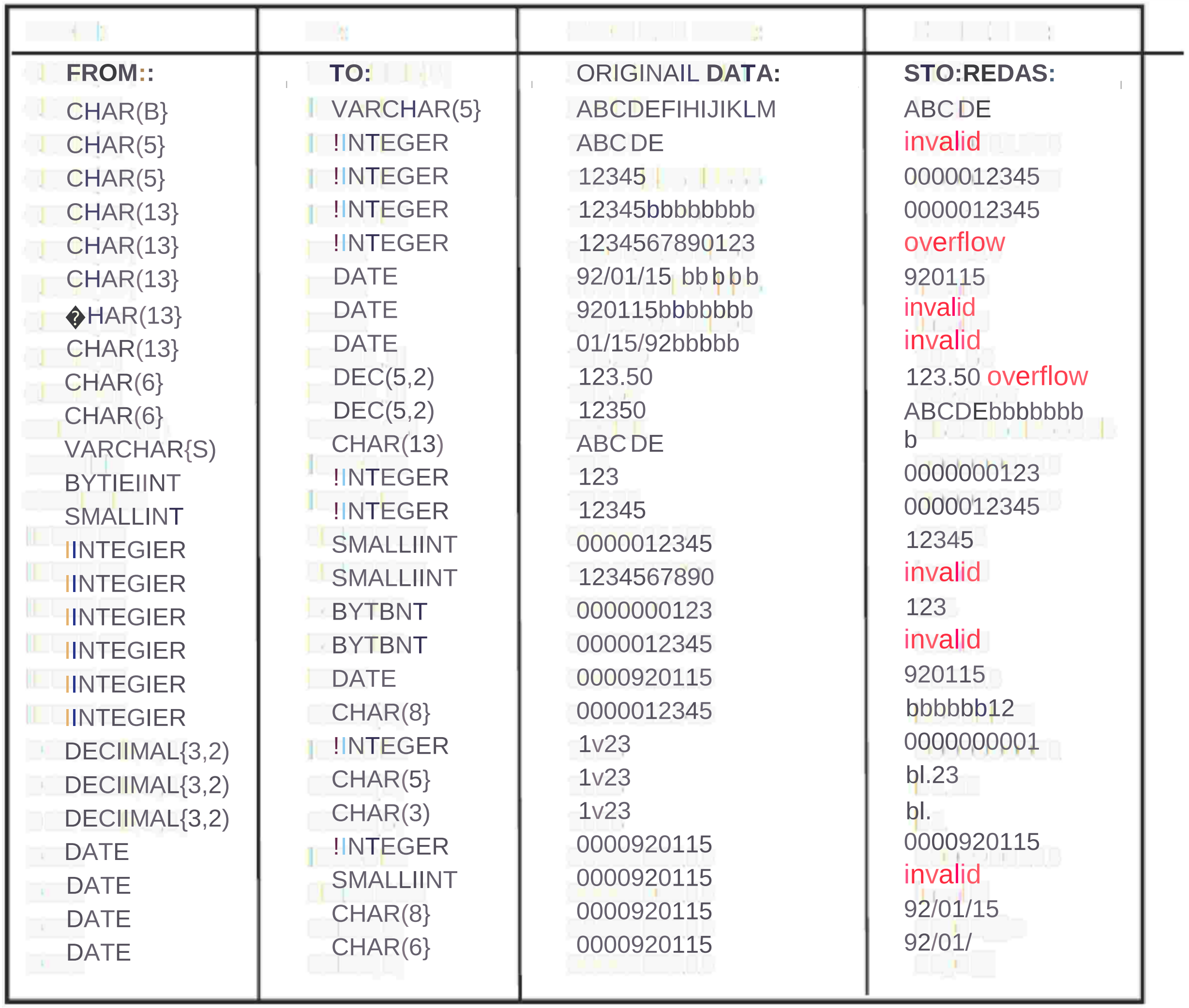
```
DEFINE FILE= data_file3;
INSERT INTO Accounts.*;
```

Data Type Conversion in Fast Load

- Converting data is easy. Just define the input data types in the input file. Then, Fast Load willll compare that to the column definitions in the Data Dictionary and convert the data for you! But the cardinal rule is that only one data type conversion is allowed per column.   n the example belllow;, notice how the columns in the input fillle are converted from one data ty[Pe to another si11mplly by redefining the data tvpe m the CREATE TABLE statement.

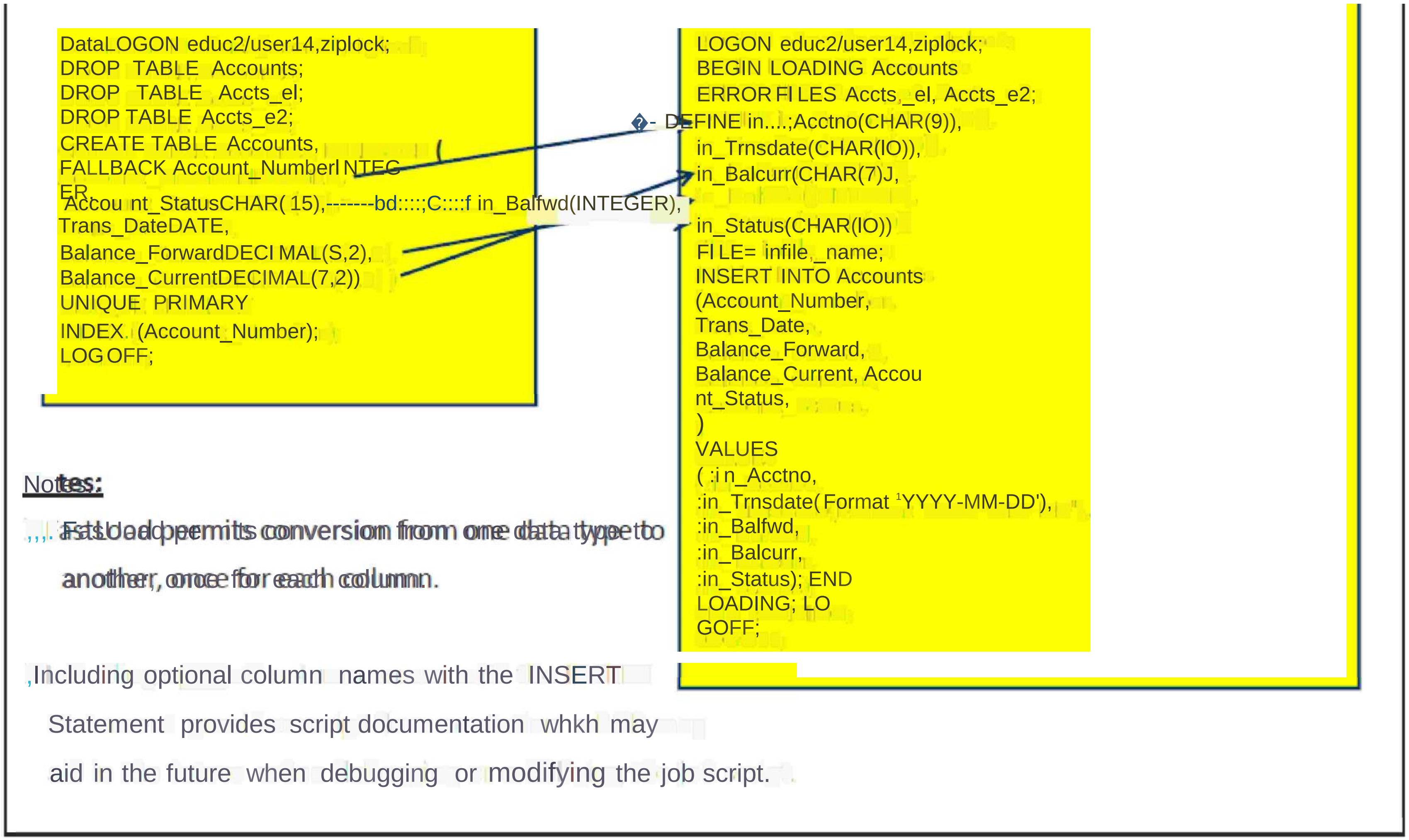
- Each input data field (DEIFINE) must undergo a conversion to fit i1n the database field (Create Ta bile).
- A.II are valllid conversions and are limited to one per column

Data Conversion Examples



FROM::	TO:	ORIGINAIL DATA:	STO:REDAS:
CHAR(B}	VARCHAR(5}	ABCDEFIHIJIKLM	ABC DE
CHAR(5}	!INTEGER	ABC DE	invalid
CHAR(5}	!INTEGER	12345	0000012345
CHAR(13}	!INTEGER	12345bbbbbbbb	0000012345
CHAR(13}	!INTEGER	1234567890123	overflow
CHAR(13}	DATE	92/01/15 bb b b b	920115
�HAR(13}	DATE	920115bbbbbbb	invalid
CHAR(13}	DATE	01/15/92bbbbb	invalid
CHAR(6}	DEC(5,2)	123.50	123.50 overflow
CHAR(6}	DEC(5,2)	12350	ABCDEbbbbbbb
VARCHAR{S)	CHAR(13)	ABC DE	b
BYTIEIINT	!INTEGER	123	0000000123
SMALLINT	!INTEGER	12345	0000012345
IINTEGIER	SMALLIINT	0000012345	12345
IINTEGIER	SMALLIINT	1234567890	invalid
IINTEGIER	BYTBNT	0000000123	123
IINTEGIER	BYTBNT	0000012345	invalid
IINTEGIER	DATE	0000920115	920115
IINTEGIER	CHAR(8}	0000012345	bbbbbb12
DECIIMAL{3,2)	!INTEGER	1v23	0000000001
DECIIMAL{3,2)	CHAR(5}	1v23	bl.23
DECIIMAL{3,2)	CHAR(3)	1v23	bl.
DATE	!INTEGER	0000920115	0000920115
DATE	SMALLIINT	0000920115	invalid
DATE	CHAR(8}	0000920115	92/01/15
DATE	CHAR(6}	0000920115	92/01/

Data type Conversion in Fast Load



```
DataLOGON educ2/user14,ziplock;
DROP TABLE Accounts;
DROP TABLE Accts_el;
DROP TABLE Accts_e2;
CREATE TABLE Accounts, (
FALLBACK Account_Numberl NTEG
ER,
Accou nt_StatusCHAR( 15),-------bd::::;C::::f in_Balfwd(INTEGER),
Trans_DateDATE,
Balance_ForwardDECI MAL(S,2),
Balance_CurrentDECIMAL(7,2))
UNIQUE PRIMARY
INDEX (Account_Number);
LOG OFF;
```

```
LOGON educ2/user14,ziplock;
BEGIN LOADING Accounts
ERROR FI LES Accts,_el, Accts_e2;
DEFINE in....;Acctno(CHAR(9)),
in_Trnsdate(CHAR(lO)),
in_Balcurr(CHAR(7)J,
in_Status(CHAR(lO))
FI LE= infile,_name;
INSERT INTO Accounts
(Account_Number,
Trans_Date,
Balance_Forward,
Balance_Current, Accou
nt_Status,
)
VALUES
( :i n_Acctno,
:in_Trnsdate( Format 'YYYY-MM-DD'),
:in_Balfwd,
:in_Balcurr,
:in_Status); END
LOADING; LO
GOFF;
```

Notes:

- FastLoad permits conversion from one data type to another, once for each column.
- Including optional column names with the INSERT Statement provides script documentation whkh may aid in the future when debugging or modifying the job script.

Fast Load Restartibilty

- When the Restart Is not Posslblec-

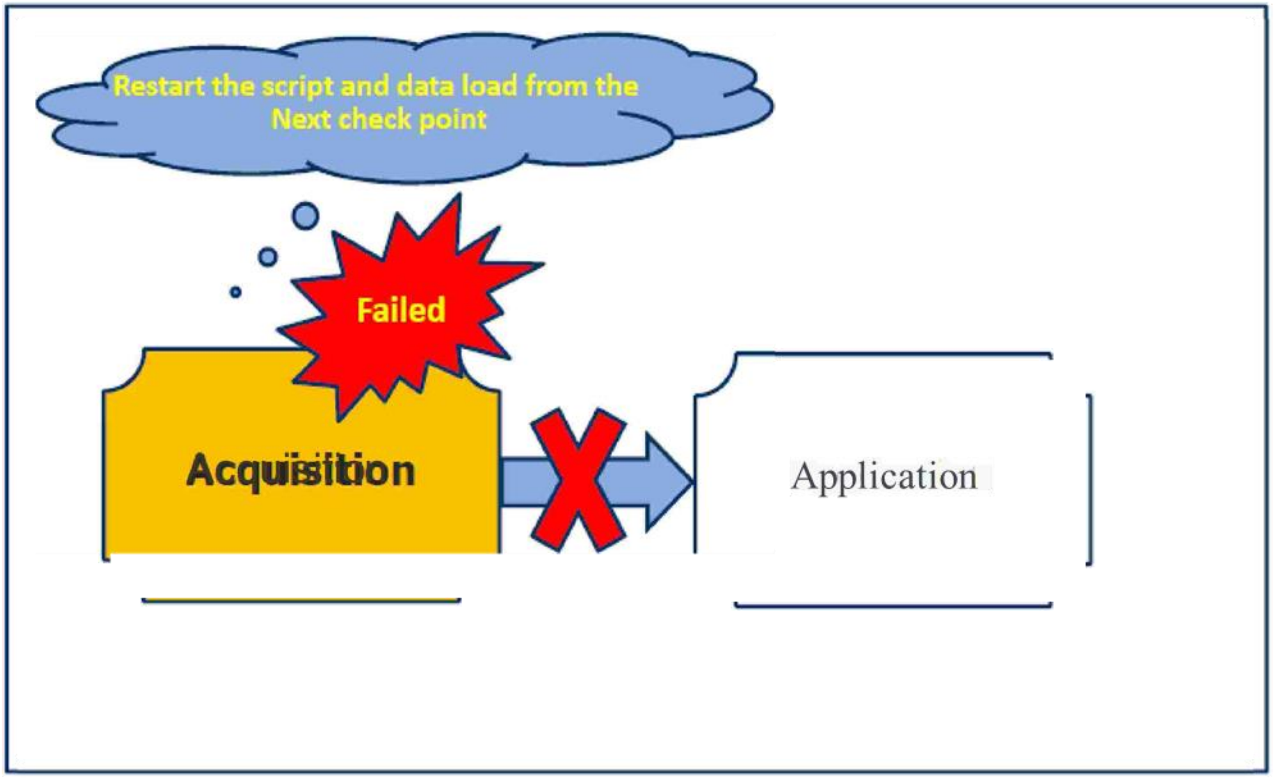
-,/' If tlhe Enror Tables are DROPPED.
·./ If the Target Table is DROPPED. If
" the Log Table is DROPPED.
·,,
,-

When the Restart ls Posslb **e:-l**

If an of the following cond itions are true, then Fast load *is Al WAYS restartable:*

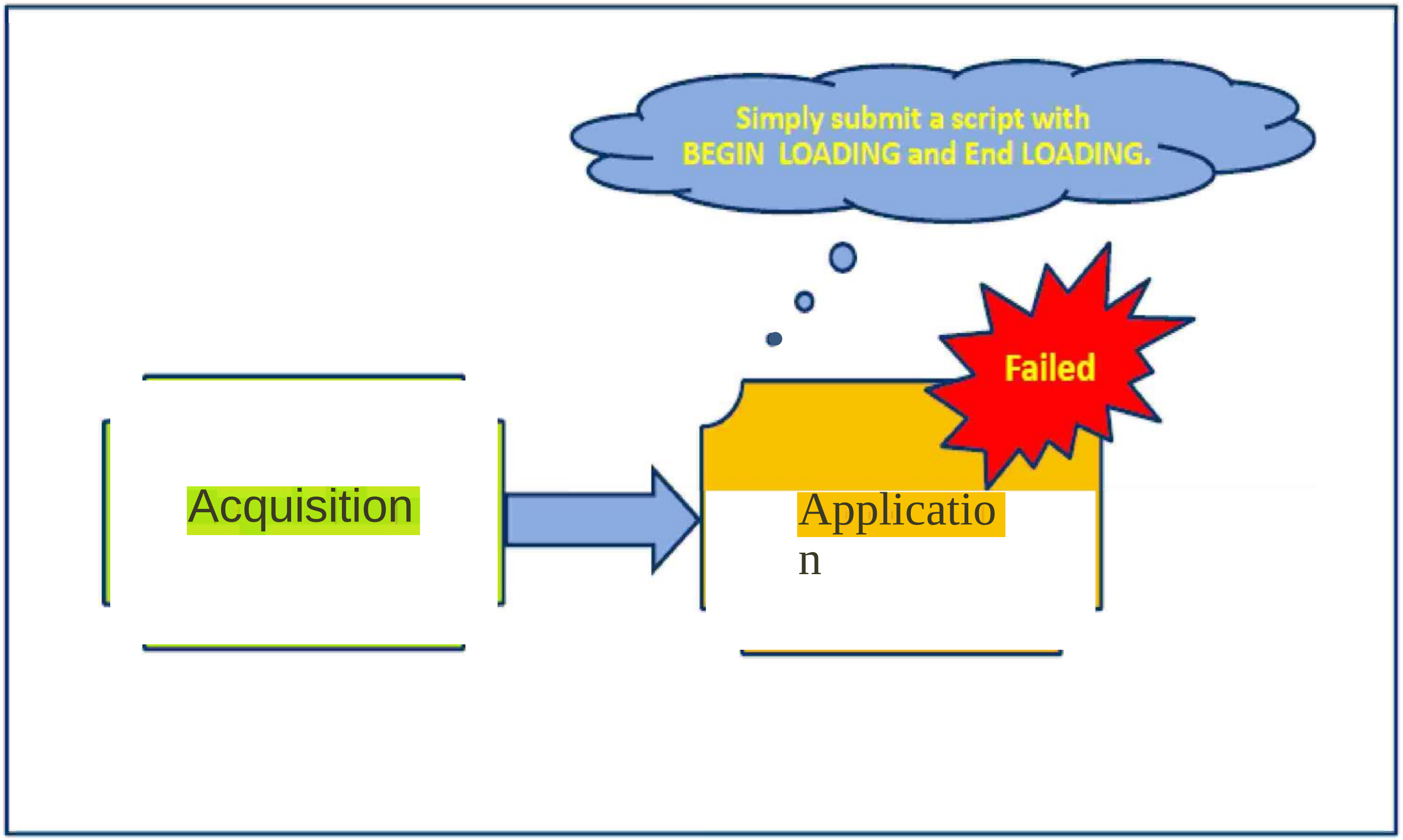
-,/' The Error Ta biles are NOT DROPPED in the script.
-,/' The Target Table is NOT DROPPED in the script.
v' The Target Table is NOT CREATED in the script,
-,/' You have defined a checkpoint.

Fast Load Restartibilty



F,ailed in A.1cqu'isit:ion Phase

Fast Load Restartibilty



Failed in A.pplication
Phase

CHECKPOINT Option

BEGIN] LOADING ....
CHECKPOINIT *integer;*

- Used to ve1ri1fy that rows have been transmrtted and processed.
- Specifies tllhe number of rows transmitted before pausingto take a checkpoint and verify receipt by AMPs.
- If the CHECKPOINT parameter is not specified, Fastload takes checkpoints as follows:
  - Beginning of Phase 1
  - Every 100.,000 **input** records
  - End **of** Phase 1
- Fastl.oari can be restarted from previous checkpoint.
- Performance Note: Checkpoints slow Fastload p,rocessi1ng- set the CHECKPO**NT** llarge enough that checkpoints are taken every 10 to 15 minutes. Usiuallly:r this requires a CHECKPOI**NT** value much larger than 100:,000.

Restarting FastLoad

Condition 1.: Abort in Phas,e 1 - data acquisition incomplete.

S,olution: Resiubmi1t the script. FastLoad wm begin from record 1 or the first record past the last checkpoint.

Condition 2: Abort occurs in Phas,e 2 - data acquisition complete,

Solution: Submit on�y BEGIN and END LOADING statements: restarts Phase 2 only.

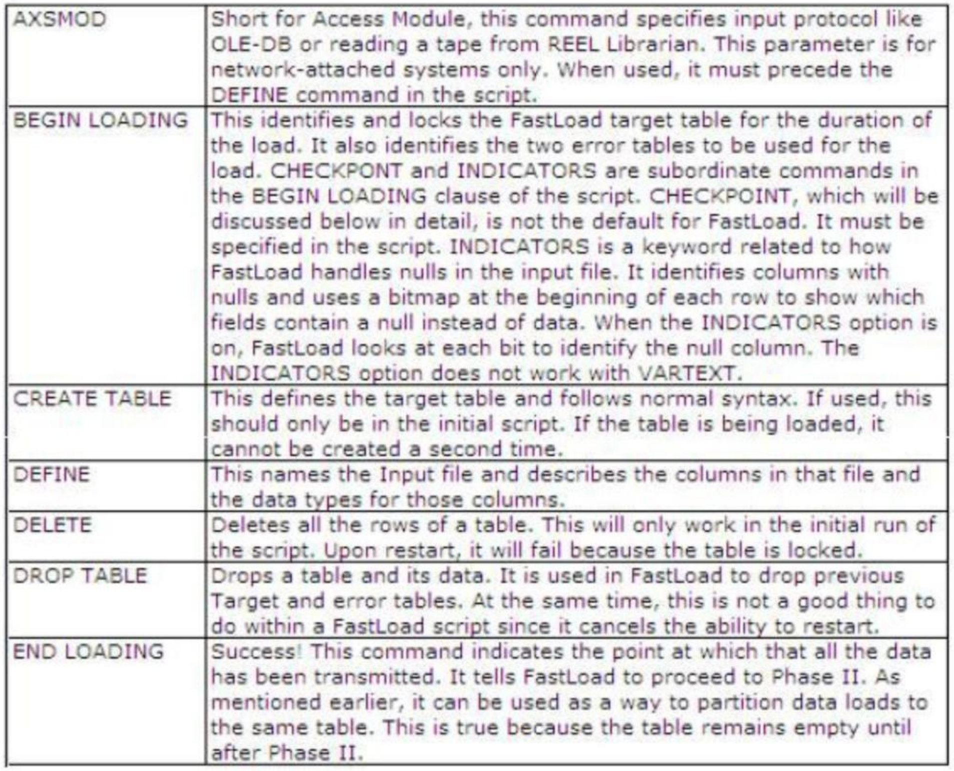
Condition .3: Normal end of Phasie l 1(paused) - more data to acquir1e�. thus there is no 'END LOADING' staternerst in scrlpt..

S,olution: Resubmit the adjusted script witllh new data file name. FastLoad wil�l be positioned to record 1 or tllhe first record past the last checkpoint.

Condition 4: Normal end of Phasie 1 (paused)-· no more data to acquir1e, no 'END LOAD NG' staternerrt was in the scrlpt..

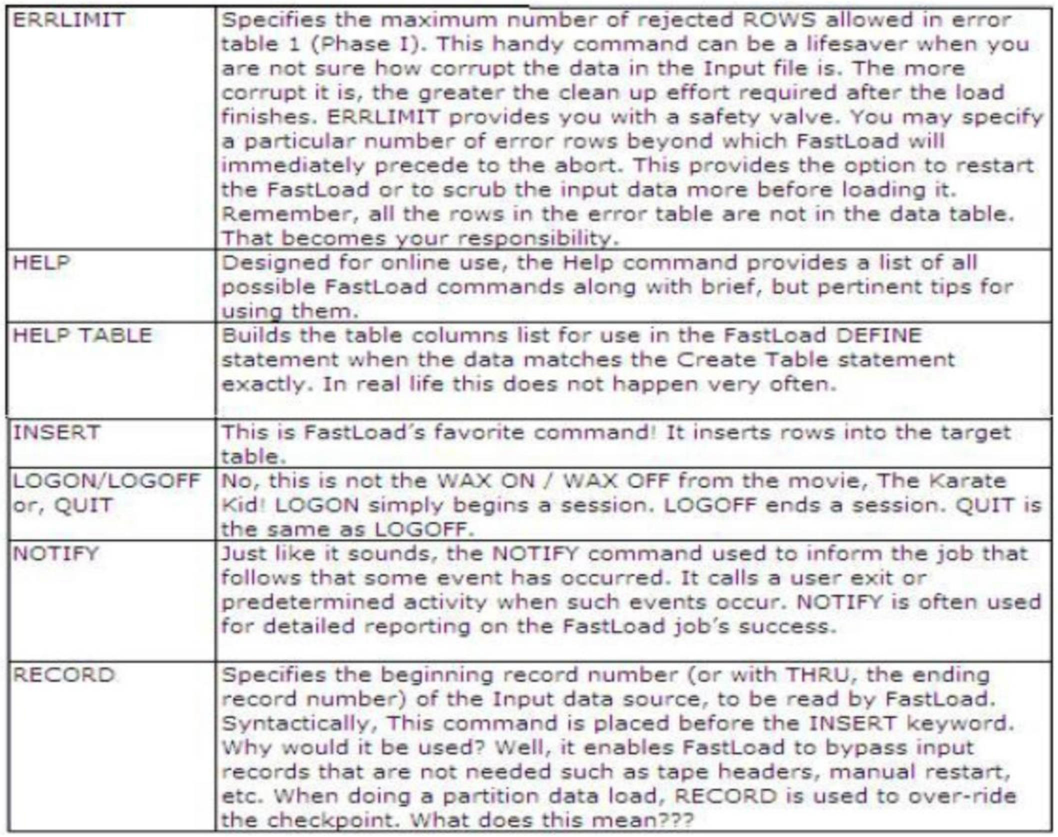
S,olution: Submit B,EGN and END LOADING statements; restarts Phase 2 only.

Fast Load Command



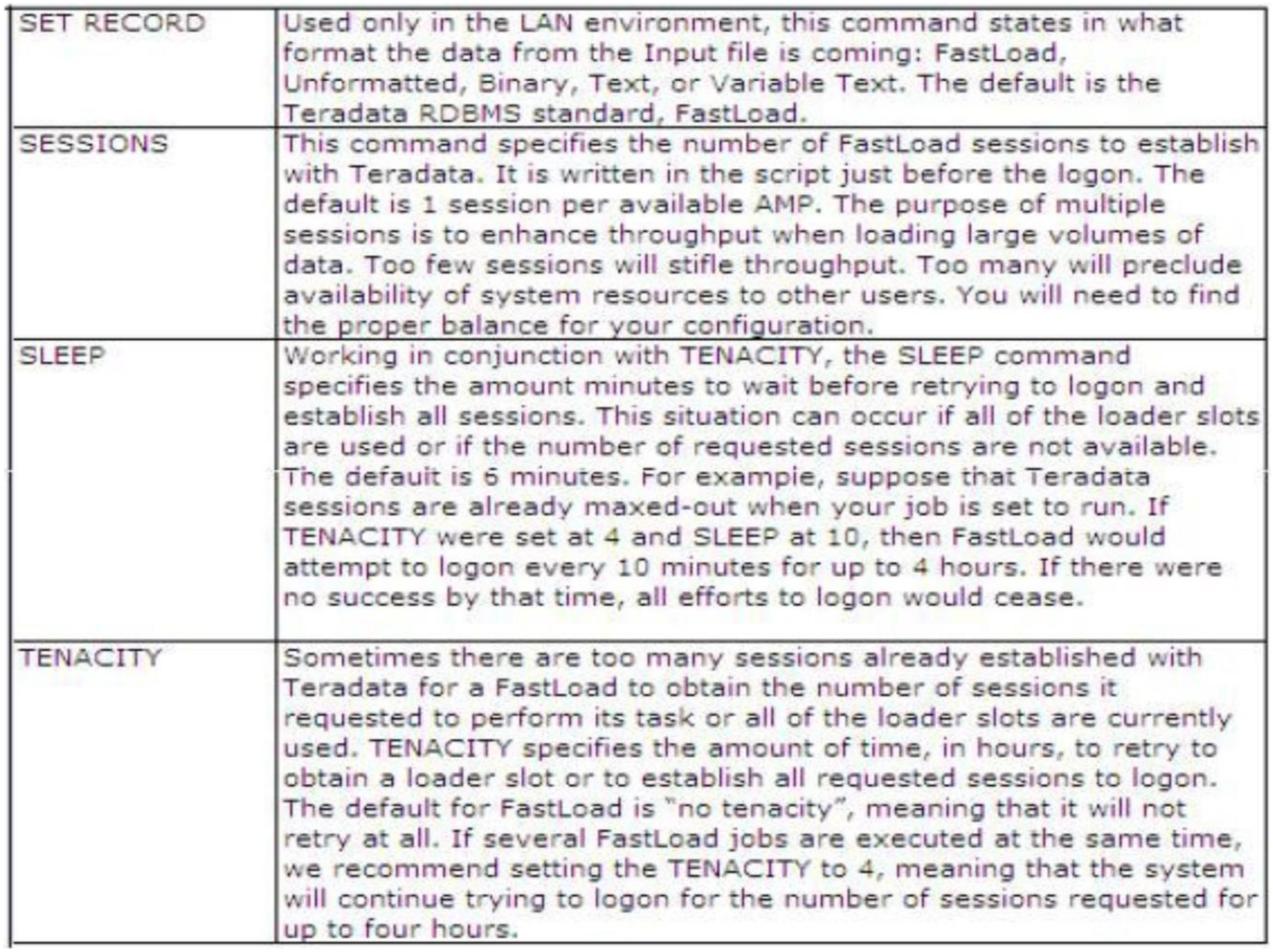
AXSMOD	Short for Access Module, this command specifies input protocol like OLE-DB or reading a tape from REEL Librarian. This parameter is for network-attached systems only. When used, it must precede the DEFINE command in the script.
BEGIN LOADING	This identifies and locks the FastLoad target table for the duration of the load. It also identifies the two error tables to be used for the load. CHECKPONT and INDICATORS are subordinate commands in the BEGIN LOADING clause of the script. CHECKPOINT, which will be discussed below in detail, is not the default for FastLoad. It must be specified in the script. INDICATORS is a keyword related to how FastLoad handles nulls in the input file. It identifies columns with nulls and uses a bitmap at the beginning of each row to show which fields contain a null instead of data. When the INDICATORS option is on, FastLoad looks at each bit to identify the null column. The INDICATORS option does not work with VARTEXT.
CREATE TABLE	This defines the target table and follows normal syntax. If used, this should only be in the initial script. If the table is being loaded, it cannot be created a second time.
DEFINE	This names the Input file and describes the columns in that file and the data types for those columns.
DELETE	Deletes all the rows of a table. This will only work in the initial run of the script. Upon restart, it will fail because the table is locked.
DROP TABLE	Drops a table and its data. It is used in FastLoad to drop previous Target and error tables. At the same time, this is not a good thing to do within a FastLoad script since it cancels the ability to restart.
END LOADING	Success! This command indicates the point at which that all the data has been transmitted. It tells FastLoad to proceed to Phase II. As mentioned earlier, it can be used as a way to partition data loads to the same table. This is true because the table remains empty until after Phase II.

Fast Load Command



ERRLIMIT	Specifies the maximum number of rejected ROWS allowed in error table 1 (Phase I). This handy command can be a lifesaver when you are not sure how corrupt the data in the Input file is. The more corrupt it is, the greater the clean up effort required after the load finishes. ERRLIMIT provides you with a safety valve. You may specify a particular number of error rows beyond which FastLoad will immediately precede to the abort. This provides the option to restart the FastLoad or to scrub the input data more before loading it. Remember, all the rows in the error table are not in the data table. That becomes your responsibility.
HELP	Designed for online use, the Help command provides a list of all possible FastLoad commands along with brief, but pertinent tips for using them.
HELP TABLE	Builds the table columns list for use in the FastLoad DEFINE statement when the data matches the Create Table statement exactly. In real life this does not happen very often.
INSERT	This is FastLoad's favorite command! It inserts rows into the target table.
LOGON/LOGOFF or, QUIT	No, this is not the WAX ON / WAX OFF from the movie, The Karate Kid! LOGON simply begins a session. LOGOFF ends a session. QUIT is the same as LOGOFF.
NOTIFY	Just like it sounds, the NOTIFY command used to inform the job that follows that some event has occurred. It calls a user exit or predetermined activity when such events occur. NOTIFY is often used for detailed reporting on the FastLoad job's success.
RECORD	Specifies the beginning record number (or with THRU, the ending record number) of the Input data source, to be read by FastLoad. Syntactically, This command is placed before the INSERT keyword. Why would it be used? Well, it enables FastLoad to bypass input records that are not needed such as tape headers, manual restart, etc. When doing a partition data load, RECORD is used to over-ride the checkpoint. What does this mean???

Fast Load Command



SET RECORD	Used only in the LAN environment, this command states in what format the data from the Input file is coming: FastLoad, Unformatted, Binary, Text, or Variable Text. The default is the Teradata RDBMS standard, FastLoad.
SESSIONS	This command specifies the number of FastLoad sessions to establish with Teradata. It is written in the script just before the logon. The default is 1 session per available AMP. The purpose of multiple sessions is to enhance throughput when loading large volumes of data. Too few sessions will stifle throughput. Too many will preclude availability of system resources to other users. You will need to find the proper balance for your configuration.
SLEEP	Working in conjunction with TENACITY, the SLEEP command specifies the amount minutes to wait before retrying to logon and establish all sessions. This situation can occur if all of the loader slots are used or if the number of requested sessions are not available. The default is 6 minutes. For example, suppose that Teradata sessions are already maxed-out when your job is set to run. If TENACITY were set at 4 and SLEEP at 10, then FastLoad would attempt to logon every 10 minutes for up to 4 hours. If there were no success by that time, all efforts to logon would cease.
TENACITY	Sometimes there are too many sessions already established with Teradata for a FastLoad to obtain the number of sessions it requested to perform its task or all of the loader slots are currently used. TENACITY specifies the amount of time, in hours, to retry to obtain a loader slot or to establish all requested sessions to logon. The default for FastLoad is "no tenacity", meaning that it will not retry at all. If several FastLoad jobs are executed at the same time, we recommend setting the TENACITY to 4, meaning that the system will continue trying to logon for the number of sessions requested for up to four hours.

Summary

- FastLoad Features and Characteristics:
  - Excellent utility for loading new tables from the host or server.
  - Loads into an empty table with no secondary indexes.
  - Can reload previously emptied tables
  - Remove referential integrity or secondary indexes prior to using fast load
  - Full Restart capability
  - Has two phases - creates an error table for each phase.
  - Error Limits and Error Tables, accessible using SQL

Review Questions

- Match the item in the first column to a corresponding statement in the second column.

1. ___ Phase 1	A. Might be used if a zero date causes an error
2. ___ CHECKPOINT	B. Table status required for loading with FastLoad
3. ___ ERRORTABLE1	C. Records written in unsorted blocks
4. ___ ERRORTABLE2	D. Records rows with duplicate values for UPI
5. ___ Empty Table	E. Not permitted on table to be loaded with FastLoad
6. ___ Secondary Index	F. Points FastLoad to a record in an input file
7. ___ Conversion	G. Can be used to restart loading from a given point
8. ___ NULLIF	H. Records constraint violations
9. ___ RECORD	I. Builds the actual table blocks for the new table
10. ___ Phase 2	J. Transform one data type to another, once per column

Review Question Answers

- Match the item in the first column to a corresponding statement in the second column.

1. _C_ Phase 1	A. Might be used if a zero date causes an error
2. _G_ CHECKPOINT	B. Table status required for loading with FastLoad
3. _H_ ERRORTABLE1	C. Records written in unsorted blocks
4. _D_ ERRORTABLE2	D. Records rows with duplicate values for UPI
5. _B_ Empty Table	E. Not permitted on table to be loaded with FastLoad
6. _E_ Secondary Index	F. Points FastLoad to a record in an input file
7. _J_ Conversion	G. Can be used to restart loading from a given point
8. _A_ NULLIF	H. Records constraint violations
9. _F_ RECORD	I. Builds the actual table blocks for the new table
10. _I__ Phase 2	J. Transform one data type to another, once per column

Lab Exercise Fast_Load 3-1

Purpose

In this lab, you will set up a restartable FastLoad operation.

What you need

There are two data file that contains customer data. You have to load those two data file into your empty customer table through two BTEQ scripts.

Tasks

2. Create a FastLoad script that loads the first 15 records (data3_1 file) and do not include the END LOADING statement in this script.
3. Create a FastLoad script that loads the additional 20 records (data3_2) and complete the FastLoad.
4. Check the result. (Your Customer table should contain 35 rows.)

Lab Solutions for Lab3-1

```
                 lla b312.f11d
LOGON    U I D/PASS:,I[)ata base;

BEGIN  LOADIING Customer
ER RDR Fi LES  cust:_errl, cust:_err2 ;
DEFINE          in  cust                (INT!EGER).
                in-lname                ,.
                in=fname                (CHAR(30)1)..
                                        (CHAR(20)1)..
   FILE=  data3_1;
INSERT  INTO Customer                   VAILUES
          (     :in_icust..
                :in   lnarne,
                :in=fname);
LOGO FF;

              fastload < lab3 12.fld
```

```
                            lab313.fM
LOGON   UID/PASS:,Database;

BEGIN  LOADIING Customer
ERROR Fi LIES  cust:_errl,  cust:_err2;
DEFINE  in   cust     (I
                      NTEGER).,.
            -  in_lname                (CIHAR(30)1)
                 in_fname              ..
                                       (CHAR(20)i)..
                             FILE=  data3_2;
!INSERT INTO Customer
   VALUES (  :in_cust..
                      :in   lname,
                      :in:)name);

                            END  LOADING;
LOGO FF;
```

fastload < lab313.fld