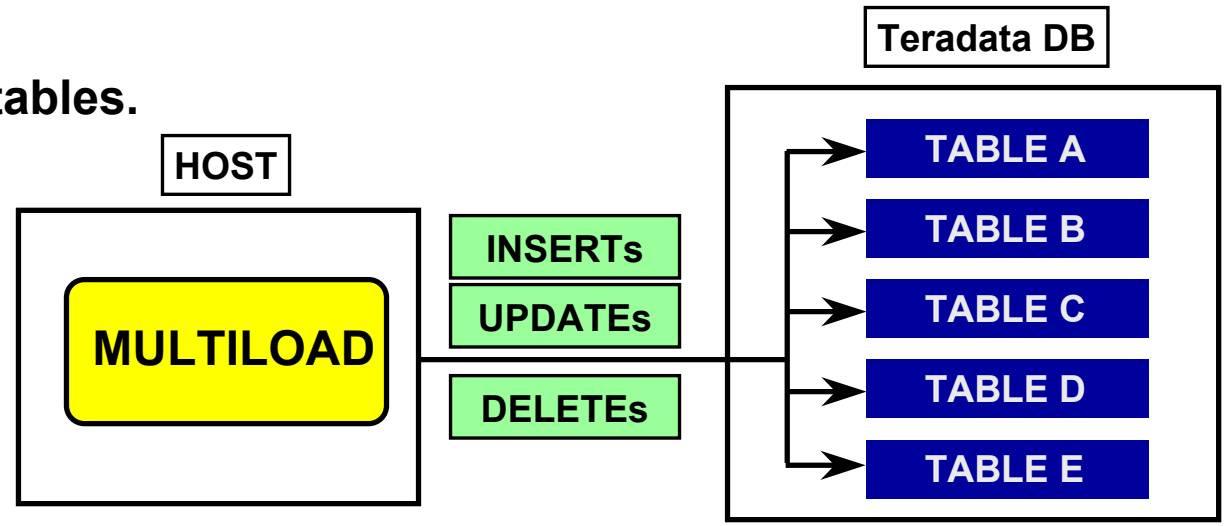
MultiLoad

After completing this module, you will be able to:

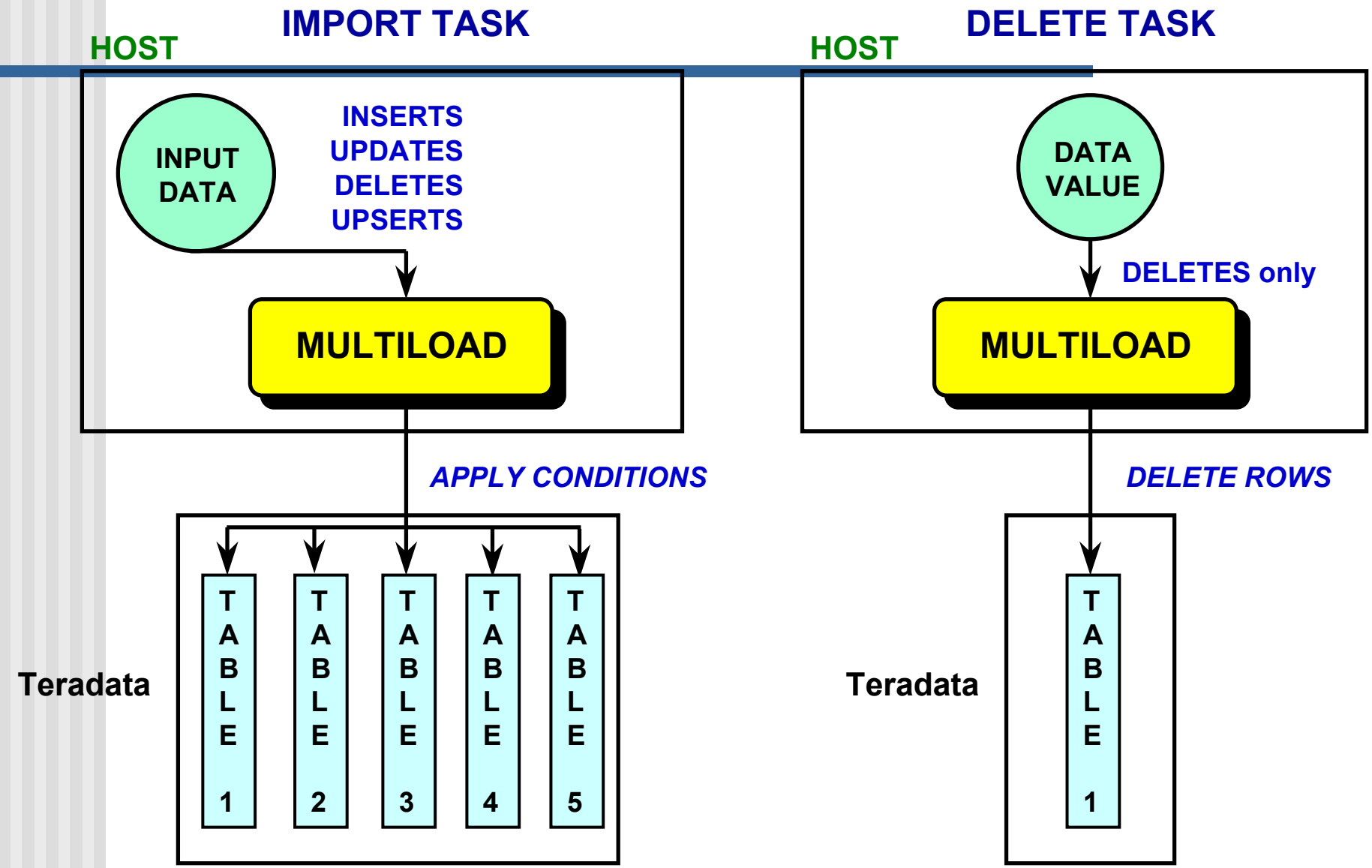
- **Describe the capabilities of MultiLoad.**
- **Name the five phases of MultiLoad and state the main function of each.**
- **Create a MultiLoad script.**
- **Run a script to update/load table(s) using MultiLoad.**
- **Explain the advantages of using MultiLoad.**

What is MultiLoad?

- Batch mode utility that runs on the host system.
- FastLoad-like technology – TPump-like functionality.
- Supports up to five populated tables.
- Multiple operations with one pass of input files.
- Conditional logic for applying changes.
- Supports INSERTs, UPDATEs, DELETEs and UPSERTs; typically with batch inputs from a host file.
- Affected data blocks only written once.
- Host and LAN support.
- Full Restart capability.
- Error reporting via error tables.
- Support for INMODs.



How MultiLoad Works



Advantages of MultiLoad

- Minimizes the use of the PEs.
- Gets input data to the AMPs as quickly as possible.
- Uses multiple-AMP sessions.
- Uses the parallelism of the AMPs to apply changes.
- Keeps BYNET activity low with AMP-local processing.
- Avoids Transient Journaling overhead.
- Allows Checkpoint/Restartability even with down AMPs.
- Prevents lengthy rollbacks of aborted jobs.
- Allows for maximum access to table during processing.
- Posts errors to special error tables.
- Provides extensive processing statistics.

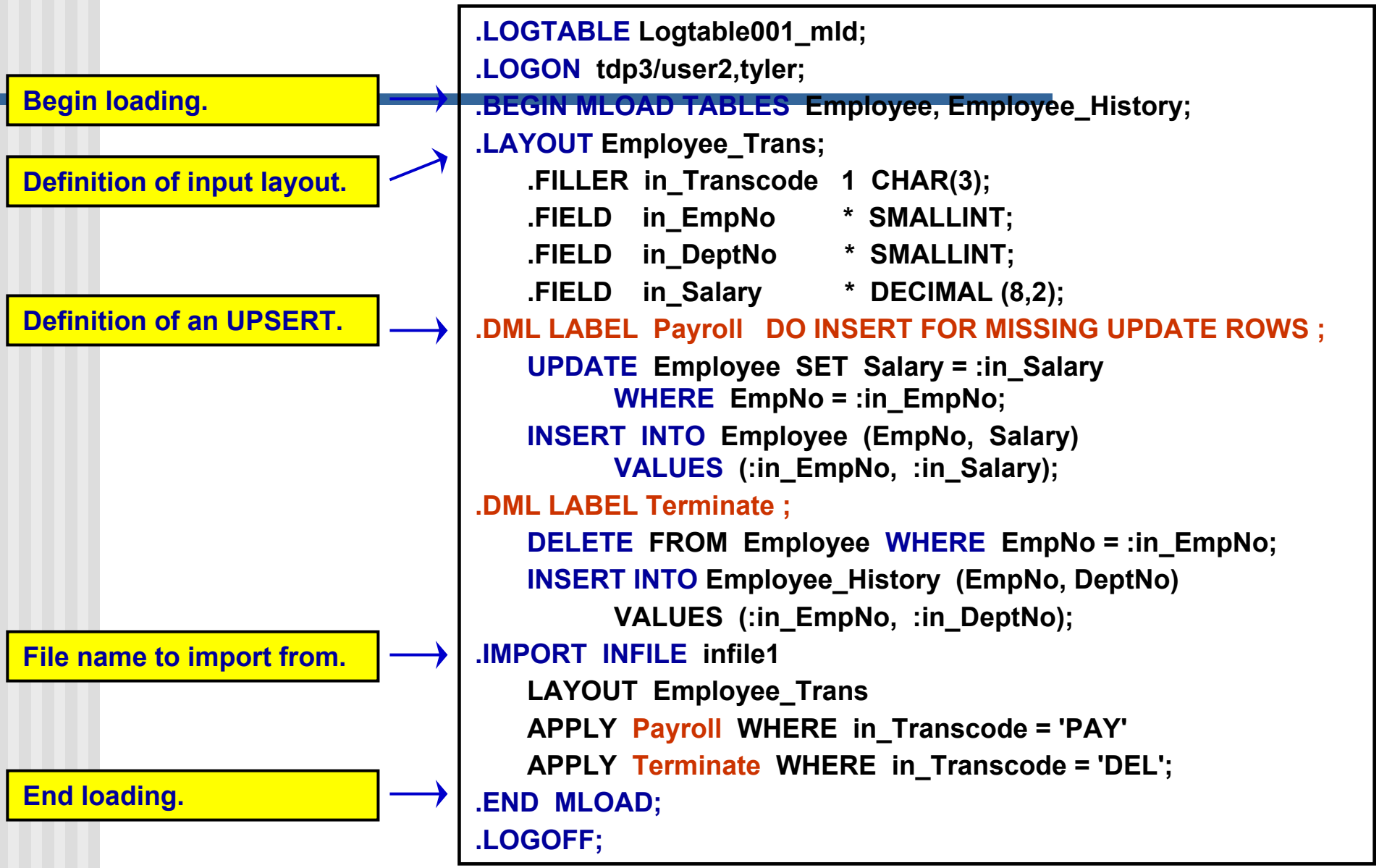
Basic MultiLoad Statements

```
.LOGTABLE  [ restartlog_tablename ] ;
.LOGON  [ tdpid/userid, password ] ;
.BEGIN MLOAD TABLES  [ tablename1, ... ] ;
.LAYOUT  [ layout_name ] ;
      .FIELD ….. ;
      .FILLER ….. ;
.DML  LABEL  [ label ] ;
.IMPORT  INFILE  [ filename ]
           [ FROM m ]   [ FOR n ]  [ THRU k ]
           [ FORMAT FASTLOAD | BINARY | TEXT | UNFORMAT |  VARTEXT 'c' ]
           LAYOUT  [ layout_name ]
           APPLY  [ label ]  [ WHERE condition ] ;
.END  MLOAD ;
.LOGOFF ;
```

```
.FIELD  fieldname { startpos datadesc } || fieldexp          [ NULLIF  nullexpr ]
    [ DROP {LEADING / TRAILING } { BLANKS / NULLS }
    [ [ AND ] {TRAILING / LEADING }  { NULLS / BLANKS } ] ] ;

.FILLER  [ fieldname ] startpos datadesc ;
```

Sample MultiLoad IMPORT Task



IMPORT Task

- **INSERTs, DELETEs, UPDATEs and UPSERTs allowed.**
- **Up to a maximum of five tables:**
  - **Empty or populated.**
  - **NUSIs permitted.**
- **MultiLoad Import task operations are always primary index operations - however, you are not allowed to change the value of a table's primary index.**
- **Change the value of a column based on its current value.**
- **Permits non-exclusive access to target tables from other users except during Application Phase.**
- **Input error limits may be specified as a number or percentage.**
- **Allows restart and checkpoint during each operating phase.**
- **IMPORT tasks cannot be done on tables with USI's, Referential Integrity, Join Indexes, Hash Indexes, or Triggers.**
  - **With V2R5, IMPORT tasks can be done on tables defined with "Soft Referential Integrity".**

5 Phases of IMPORT Task

Preliminary	Basic set up
DML Transaction	Send the DML steps to the AMPs
Acquisition	Send the input data to the AMPs
Application	Apply the input data to appropriate table(s)
Cleanup	Basic clean up

Details

Phase 1: Preliminary

IMPORT Phases

Validate all statements	→	MultiLoad and SQL
Start all sessions	→	#AMPS + 2
Create work tables	→	One per target table
Create error tables	→	Two per target table
Create Restart log	→	One per IMPORT run
Apply locks to target tables	→	Prevent DDL

Phase 2: DML Transaction

IMPORT Phases

Send prototype DML to the Teradata Database

→ **Store DML steps in work tables**

Add a USING modifier to the request

→ **Host data to be filled in from input file**

Add a "Match Tag" to the request

→ **Allows link between DML and transaction record**

Phase 3: Acquisition

IMPORT Phases

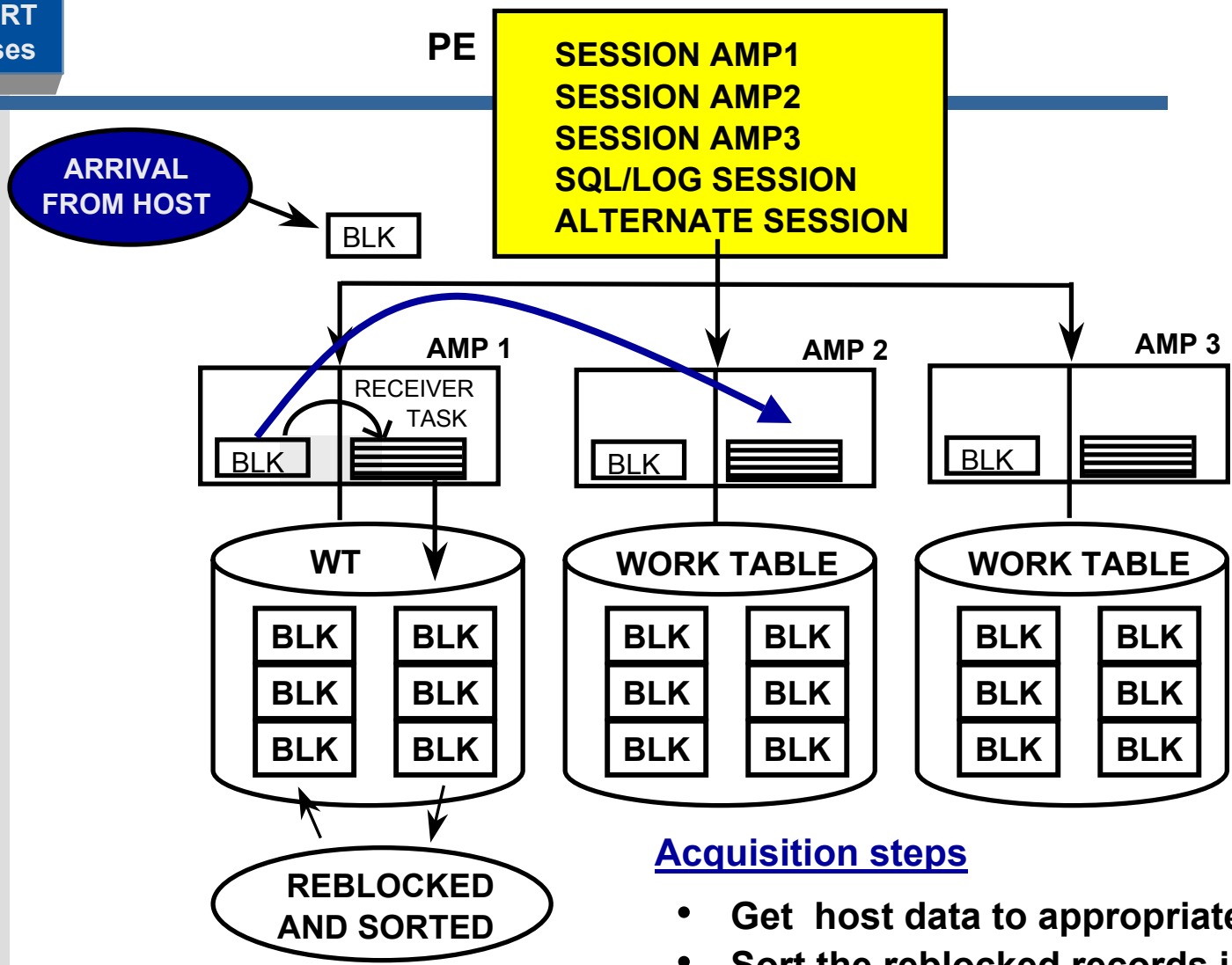
- **Get the data from host and apply it to appropriate AMP worktables.**
  - Duplicate "input records" record for each successful APPLY.
  - Add "Match Tag" information to record.
  - Make blocks and send "quickpath" to AMPs.
  - Deblock and resend record to "correct" AMP.
- **Reblock and store in worktable of target table.**
  - Sort the reblocked records in the work tables.
  - Sort by hash value and sequence to be applied.
- **Set up transition to the Application phase.**
  - Upgrade locks on target tables to Write.
  - Set table headers for Application phase.
  - This is effectively the "point of no return".

Notes:

- Errors that occur in this phase go into the Acquisition Error Table (default name is ET_tablename).
- There is no acquisition phase activity for a DELETE Task.

Phase 3: Acquisition – a Closer Look

IMPORT Phases



Acquisition steps

- Get host data to appropriate AMP worktables.
- Sort the reblocked records in the work tables.
- Set up transition to the Application phase.

Phase 4: Application

IMPORT Phases

- **Execute MLOAD for *each* target table as a *single* multi-statement request.**
  - **End of host interaction until end of phase.**
  - **AMPs independently apply changes to target tables.**
  - **Executed as a single transaction without rollback.**
  - **Restartable based on last checkpoint.**
  - **No transient journal needed.**

Note:

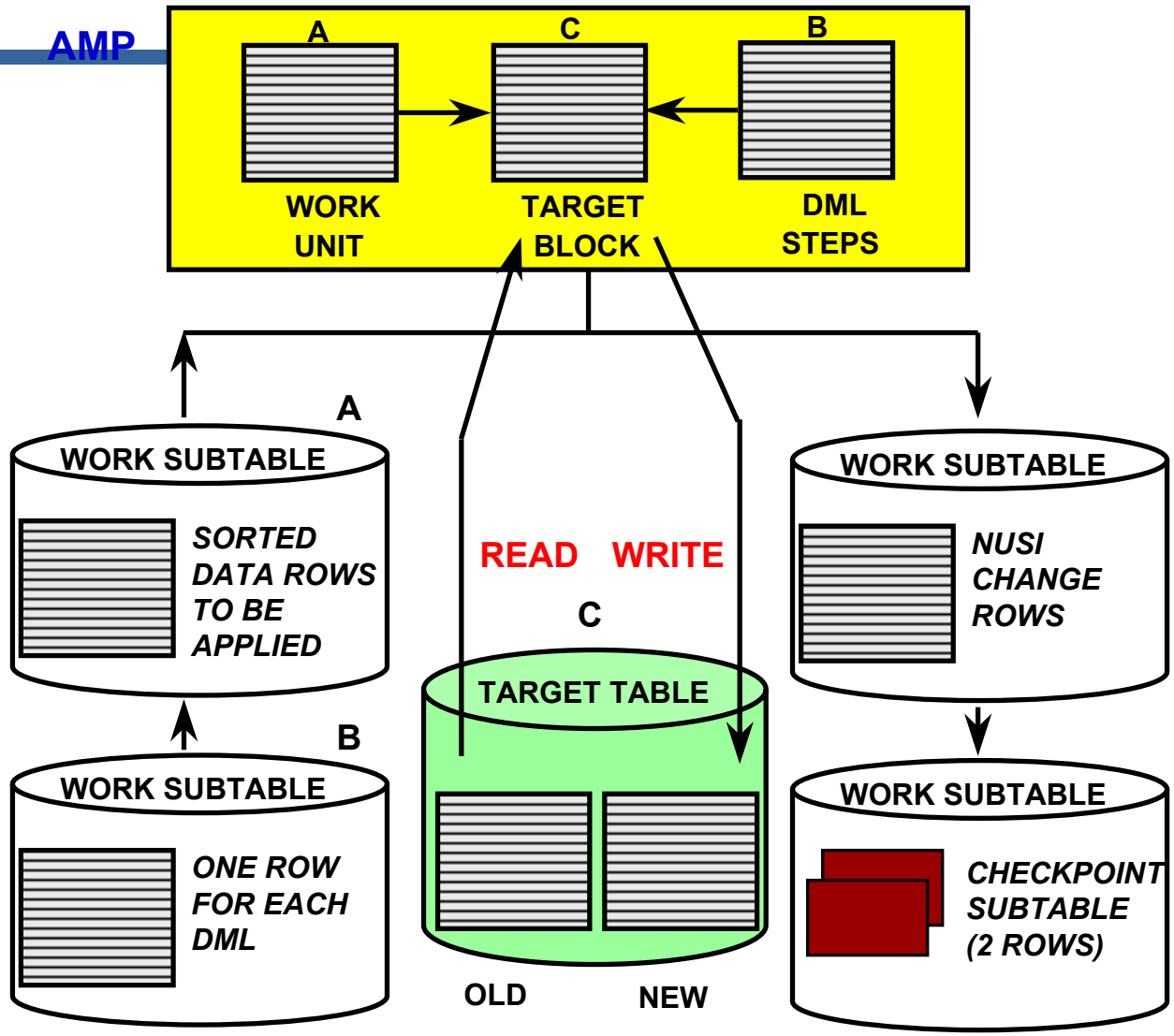
- **Errors that occur in this phase go into the Application Error Table (default name is UV_tablename).**

Phase 4: Application – a Closer Look

IMPORT Phases

Apply work subtable changes to target subtables:

- Affected blocks read/written only once.
- Changes applied based on matching row-hash.
- Errors written to appropriate error table.
- Checkpoint after writing each block.
- NUSI subtable changes applied.



Phase 5: Cleanup

IMPORT Phases

- **Execute END MLOAD processing as a series of transactions performed by the host utility:**
  - All locks are released.
  - Table headers are restored across all AMPs.
  - Dictionary cache of Target Tables is spoiled.
  - Statistics are reported.
  - Final Error Code is reported.
  - Target tables are made available to other users.
  - Work Tables are dropped.
  - Empty Error Tables are dropped.
  - Log Table is dropped (if Error Code = 0).
- **MLOAD Session Logoff:**
  - LOGOFF request is sent to each AMP with a session.

Sample MultiLoad DELETE Tasks

Hard code the values of rows to be deleted.

→

```
.LOGTABLE  Logtable002_mld;
.LOGON     tdp3/user2,tyler;
.BEGIN  DELETE  MLOAD TABLES Employee;
DELETE FROM  Employee WHERE Term_date  >  0;
.END  MLOAD;
.LOGOFF;
```

Pass a single row containing value(s) to be used.

→

```
.LOGTABLE Logtable003_mld;
.LOGON  tdp3/user2,tyler;
.BEGIN DELETE MLOAD TABLES Employee;
.LAYOUT Remove;
     .FIELD in_Termdate   *  INTEGER;
DELETE FROM  Employee  WHERE Term_date  >  :in_Termdate;
.IMPORT  INFILE  infile2
     LAYOUT  Remove;
.END  MLOAD;
.LOGOFF;
```

DELETE Task Differences from IMPORT Task

DELETE tasks operate very similarly to IMPORT tasks with some differences:

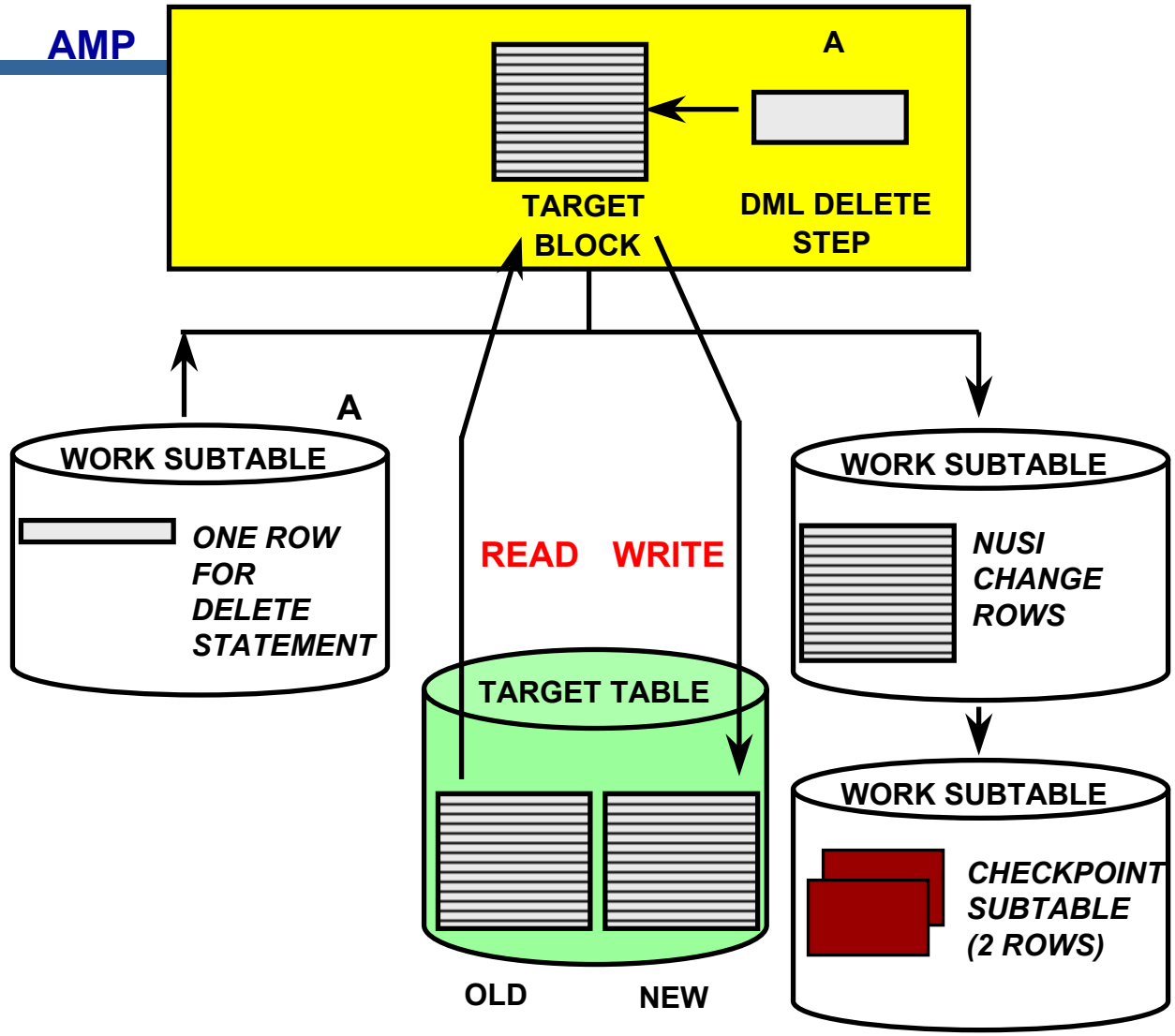
- **Deleting based on a UPI access is not permitted.**
- **A DML DELETE statement is sent to each AMP with a match tag parcel.**
- **No Acquisition phase because no variable input records to apply.**
- **Application phase reads each target block and deletes qualifying rows.**
- **All other aspects similar to IMPORT task.**

Why use MultiLoad DELETE (versus SQL DELETE)?

- **MultiLoad DELETE is faster and uses less disk disk (no Transient Journal).**
- **MultiLoad DELETE is restartable.**
  - **If SQL DELETE is aborted, Teradata applies Transient Journal rows. SQL DELETE can be resubmitted, but starts from beginning.**

A Closer Look at DELETE Task Application Phase

- **Note absence of Work Table for Import rows.**
- **Faster than traditional SQL DELETE due to:**
  - **Lack of transient journaling**
  - **No rollback of work**
  - **Restartable from checkpoint**



MultiLoad Locks

Utility locks: Placed in table headers to alert other utilities that a MultiLoad is in session for this table. They include:

- **Acquisition lock**
  - **DML — allows all**
  - **DDL — allows DROP only**
- **Application lock**
  - **DML — allows SELECT with ACCESS only**
  - **DDL — allows DROP only**

Restarting MultiLoad

Teradata Restart

- **MLOAD reinitiated automatically after Teradata recovery.**
- **Continue from checkpoint without user interaction.**

Host restart

- **Resubmit the original script.**
- **MLOAD determines its stopping point and restarts.**

MultiLoad Checkpointing Strategy

Acquisition phase

- **Checkpointing is performed according to user.**
- **Checkpoint based on time or on number of records.**
- **Default checkpoint interval is fifteen minutes.**

Application phase

- **Checkpointing done after each write of data block.**
- **Each block is written at most only one time.**

Sort phase(s)

- **Sort operations do their own internal checkpointing.**

RELEASE MLOAD Statement

RELEASE MLOAD Employee, Job, Department;

- **Returns target tables to general availability.**
- **Prevents any attempt to restart MultiLoad.**
- **Cannot be successful in all cases.**
- **Cannot override a target table lock.**
- **IMPORT — possible before Application phase.**
- **DELETE — possible during Preliminary phase.**

To successfully complete a RELEASE MLOAD:

1. **Make sure MLOAD is not running; abort if it is. (If it is past the point of no return, go to step 4.)**
2. **Enter RELEASE MLOAD.**
3. **If successful, drop the log, work, and error tables.**
4. **If not successful:**
   - **a.) restart MLOAD and let it complete, or**
   - **b.) drop target, work, and error tables, or**
   - **c.) handle error as appropriate.**

Invoking MultiLoad

Network Attached Systems: mload [PARAMETERS] < *scriptname* >*outfilename*

Channel-Attached MVS Systems: // EXEC TDSMLOAD MLPARM= [PARAMETERS]

Channel-Attached VM Systems: EXEC MLOAD [PARAMETERS]

Channel Parameter	Network Parameter	Description
BRIEF	-b	Reduces print output runtime to the least information required to determine success or failure.
CHARSET=*charsetname*	-c *charsetname*	Specify a character set or its code. Examples are EBCDIC, ASCII, or Kanji sets.
ERRLOG=*filename*	-e *filename*	Alternate file specification for error messages; produces a duplicate record.
"*multiload command*"	-r '*multiload cmd*'	Signifies the start of a MultiLoad job; usually a RUN FILE command that specifies the script file.
MAXSESS=*max sessions*	-M *max sessions*	Maximum number of MultiLoad sessions logged on.
MINSESS=*min sessions*	-N *min sessions*	Minimum number of MultiLoad sessions logged on.
< *scriptname*		Name of file that contains MultiLoad commands and SQL statements.
> *outfilename*		Name of output file for MultiLoad messages.

Application Utility Checklist

Feature	BTEQ	FastLoad	FastExport	MultiLoad	TPump
DDL Functions	ALL	LIMITED	No	ALL	
DML Functions	ALL	INSERT	SELECT	INS/UPD/DEL	
Multiple DML	Yes	No	Yes	Yes	
Multiple Tables	Yes	No	Yes	Yes	
Multiple Sessions	Yes	Yes	Yes	Yes	
Protocol Used	SQL	FASTLOAD	EXPORT	MULTILOAD	
Conditional Expressions	Yes	No	Yes	Yes	
Arithmetic Calculations	Yes	No	Yes	Yes	
Data Conversion	Yes	1 per column	Yes	Yes	
Error Files	No	Yes	No	Yes	
Error Limits	No	Yes	No	Yes	
User-written Routines	No	Yes	Yes	Yes	

Summary

- **Batch mode utility.**
- **Supports up to five populated tables.**
- **Multiple operations with one pass of input files.**
- **Conditional logic for applying changes.**
- **Supports INSERTs, UPDATEs, DELETEs and UPSERTs; typically with batch inputs from a host file.**
- **Affected data blocks only written once.**
- **Full Restart capability.**
- **Error reporting via error files.**
- **Support for INMODs.**

Review Questions

Answer True or False.

1. True or False. With MultiLoad, you can import data from the host into populated tables.
2. True or False. MultiLoad cannot process tables with USIs or Referential Integrity defined.
3. True or False. MultiLoad allows changes to the value of a table's primary index.
4. True or False. MultiLoad allows you to change the value of a column based on its current value.
5. True or False. MultiLoad permits non-exclusive access to target tables from other users except during Application Phase.

Match the MultiLoad Phase in the first column to its corresponding task in the second column.

1. ___ Preliminary	A. Acquires or creates Restart Log Table.
2. ___ DML Transaction	B. Locks are released.
3. ___ Acquisition	C. Applies (loads) data to the work tables.
4. ___ Application	D. Execute mload for each target table as a single multi-statement request.
5. ___ Cleanup	E. Stores DML steps in work tables

Review Question Answers

Answer True or False.

1. ***True*** or False. With MultiLoad, you can import data from the host into populated tables.
2. ***True*** or False. MultiLoad cannot process tables with USIs or Referential Integrity defined.
3. True or ***False***. MultiLoad allows changes to the value of a table's primary index.
4. ***True*** or False. MultiLoad allows you to change the value of a column based on its current value.
5. ***True*** or False. MultiLoad permits non-exclusive access to target tables from other users except during Application Phase.

Match the MultiLoad Phase in the first column to its corresponding task in the second column.

1. _*A*_ Preliminary	A. Acquires or creates Restart Log Table.
2. _*E*_ DML Transaction	B. Locks are released.
3. _*C*_ Acquisition	C. Applies (loads) data to the work tables.
4. _*D*_ Application	D. Execute mload for each target table as a single multi-statement request.
5. _*B*_ Cleanup	E. Stores DML steps in work tables

Lab Exercises

Lab Exercise

Purpose

In this lab, you will use MultiLoad to delete rows from your three tables. An input file will be created which will contain a control letter (A - Accounts, C - Customer, and T - Trans) followed by a primary index value for the appropriate table.

What you need

Your three tables with two hundred (200) rows in each.

Tasks

1. **Prepare the data file by executing the macro AU.Lab6_1. Export your data to a file called *data6_1*.**

2. **Prepare your tables by doing the following:**

   a. **In BTEQ, issue a Delete All command on each of your tables.**

   b. **While still in BTEQ, execute the following script which will load the specified 200 rows into each of the tables:**

```
INSERT INTO Accounts    SELECT * FROM AU.Accounts    WHERE Account_Number LT 20024201;

INSERT INTO Customer    SELECT * FROM AU.Customer    WHERE Customer_Number LT 2201;

INSERT INTO Trans       SELECT * FROM AU.Trans       WHERE Account_Number LT 20024201;
```

3. **Prepare your MultiLoad script to Delete Rows from each of the tables depending on the incoming code (A, C, or T) from *data6_1*. This job should result in deleting 100 rows from each of the three tables.**

4. **Check your results by doing a SELECT COUNT(*) on each of your tables.**

Lab Solution for Lab

cat lab613.mld

```
.LOGTABLE Restartlog613_mld;
.LOGON u4455/tljc30,tljc30 ;
.BEGIN MLOAD TABLES Accounts, Customer, Trans ;

.LAYOUT  Record_Layout_613;
      .FILLER    File_Control      *      CHAR(1) ;
      .FIELD     PI_Value          *      INTEGER ;

.DML LABEL Del_Acct ;
      DELETE FROM Accounts    WHERE Account_Number = :PI_Value ;

.DML LABEL Del_Cust ;
      DELETE FROM Customer    WHERE Customer_Number = :PI_Value ;

.DML LABEL Del_Trans ;
      DELETE FROM Trans       WHERE Account_Number = :PI_Value ;

.IMPORT INFILE data6_1
      LAYOUT Record_Layout_613
      APPLY Del_Acct          WHERE File_Control = 'A'
      APPLY Del_Cust          WHERE File_Control = 'C'
      APPLY Del_Trans         WHERE File_Control = 'T' ;

.END MLOAD ;
.LOGOFF ;
```

mload < lab613.mld > lab613.out