Chapter 11
SQL*Loader

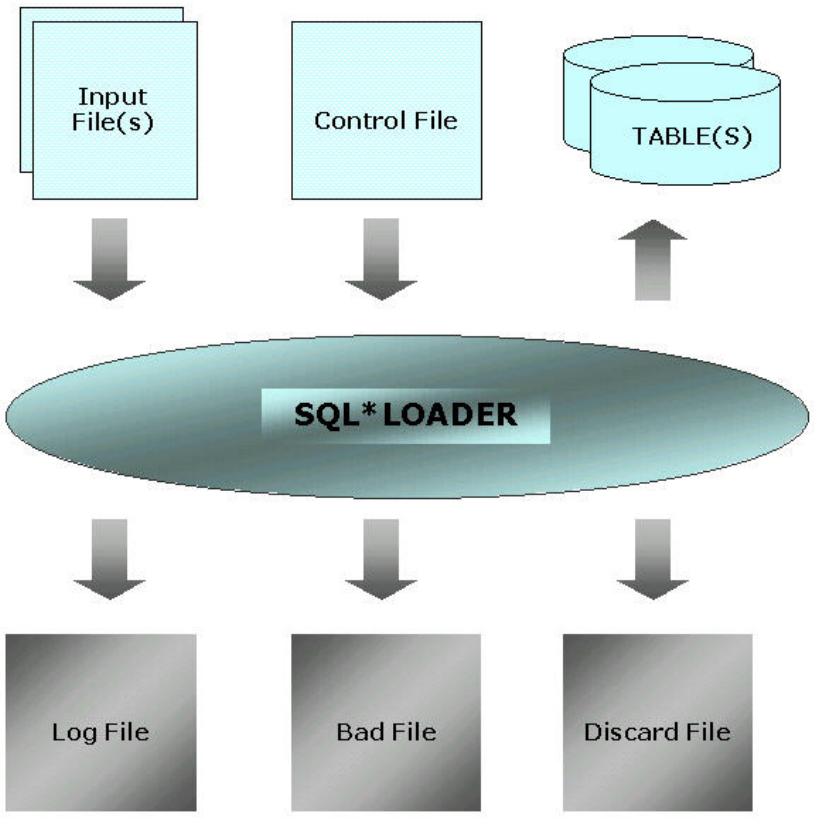
SQL *Loader is a utility program that will read flat files and insert the data they contain into database tables as specified by a **control file**. The Control File is a file you write specifying the name(s) of the input file(s), the name(s) of the table(s), and other parameters specifying the file format, etc. SQL*Loader will accept data in either fixed or variable record length formats. A file referenced in the control file may contain data that may be inserted into multiple tables.

SQL*Loader is useful if you receive electronic submissions of data from different sources and/or in different formats. It is especially useful to load external data into a data warehouse. It allows you to check data from validity and filter out data you may not want inserted into the tables. You can add rows to existing tables or totally replace the data in an existing table. SQL*Loader is run from the host system prompt, not from within SQL unless you use ! to access the operating system.

SQL*Loader Files

Figure 1 illustrates the files used by SQL*Loader.

Figure 1
SQL*Loader File And Table Relations



The table below describes each of the files depicted in the diagram above.

File	***Description***
INPUT	The Input File contains the data that you would like to load into the tables. You can specify more than one Input File in the Control File. The Input File may contain records in either a fixed or variable format.
CONTROL	The Control File specifies the Input File(s) and Table(s) to be used in the load. The Control File also specifies several parameters that tell SQL*Loader the record format, items to be filtered, and how the data are to be added to the table. The precise format of the Control File is shown below and several examples are illustrated later in this chapter.
LOG	The Log File provides a summary of the SQL*Load describing records loaded, records rejected, and errors that occurred. It is an output file created by SQL*Loader.
BAD	The Bad File is an output file created by SQL*Loader containing records that could not be added to the table because either the format of the record wasn't valid or the record violated an integrity constraint placed on the table.
DISCARD	The Discard File is an output file created by SQL*Loader that contains records that violate a filter defined in the Control File.

The Control File

The Control File is created in the operating system with a text editor such as **pico** or **vi**. The file should be given a name that relates it to the table(s) you are loading. If you are loading the EMP table with a fixed file format, you might call the control file **emp_fixed.ctl**. The structure of the Control File is shown below. The items you enter into the control file are shown in bold. The items in bold italics require you to substitute an actual value for the item.

PATHS

OPTIONS (DIRECT = TRUE, SKIP = *N*, LOAD = *N*, ERRORS = *N*)

- The Conventional path is the default. It loads rows using all safeguards. The Direct Path does not enforce all table integrity constraints but loads data much faster than the conventional path.
- Supplying a value for N in Skip will skip n number of rows in the Infile.
- Supplying a value for N in Load will load n records from Infile.
- Supplying a value for N in Errors will cause SQL*Loader to terminate if n errors are generated during the load.

Unrecoverable (writing to the REDO Log is turned off (Direct Only))

LOAD DATA

Supply the names of the Input File (Infile), the Bad File and the Discard File. You are required to provide an Infile name. If you don't want to save the records that are rejected, do not use Badfile or Discardfile.

Infile ***filename***
Badfile ***filename*** (optional)
Discardfile ***filename*** (optional)

The Loading Methods available are: INSERT, APPEND, REPLACE, or TRUNCATE. If you use INSERT, the table must not contain rows or an error will be generated. If you use APPEND, rows will be added to an existing table. REPLACE and TRUNCATE are similar to one another as both delete all data from a table and then insert the data defined in Infile. TRUNCATE will deallocate extents after deleting data but DELETE will not.

INTO TABLE ***tablename***

Use the WHEN condition to filter out data you don't want inserted into the table. The WHEN condition is used to catch rows that have a valid format and don't violate any database constraints, but you still don't want the data loaded.

WHEN ***condition*** (optional)

For fixed record data file formats use the following structure. Replace actual column names in the table with Col1, Col2, etc. The data for the columns should be located at certain column positions within the Infile. In position(x,y) statement, the x refers to the starting column position and the y refers to the ending column position. Normally you would specify the data type as CHAR even if the data type in the table is different. The database will properly convert the data value from the Infile.

```
(Col1        position(x,y)    char or integer external or date,
 Col2        position(x,y)    char or integer external or date,
 etc.
)
```

For variable record data file formats you must have a character (or characters) that separate data items. Writing the column statements for variable record formats is shown below with fields separated by either a comma or "Whitespace".

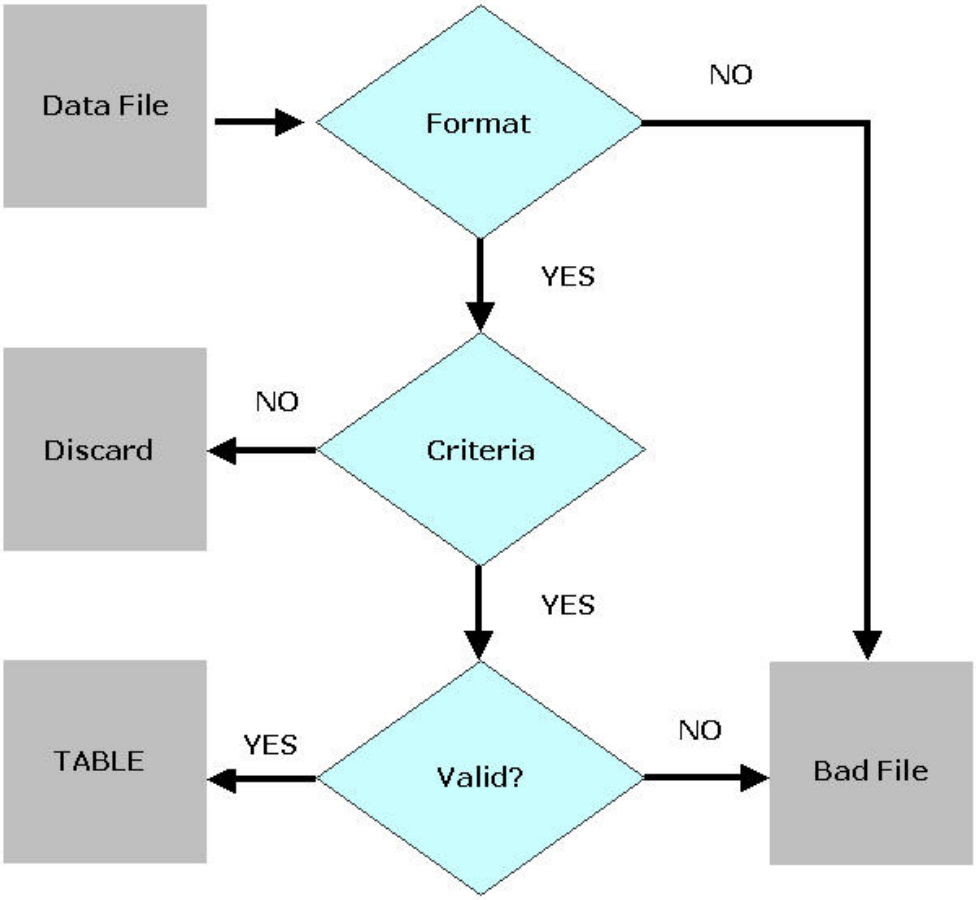
Fields Terminated by "," or Whitespace

```
(Col1          char  or  integer external  or  date,
 Col2          char  or  integer external  or  date,
 Col3          char  or  integer external  or  date
 )
```

SQL*Loader Data Flows

SQL*Loader reads the input data file and first checks it for a valid format. If an error is found in the format (for example a character is found where a number should be) the record is inserted into the Bad File. If the record has a valid format the WHEN condition is checked. If the record fails the WHEN condition, it is written to the Discard File. Finally, the record is checked against integrity constraints enforced on the table. If the record is not valid, it is written to the Bad File. If the record meets all criteria, it is inserted into the table. Figure 2 shows the flow of a record in SQL*Loader.

Figure 2
Data Flow For SQL*Loader



Executing SQL*Loader

SQL*Loader is run from the unix prompt. You must supply a login and a password for an account with privileges to use SQL*Loader as shown below:

```
[24]> sqlldr userid=system/manager
```

Press enter and you will be prompted for a control file. Enter the name of the control file and SQL*Loader will execute.

If you are in SQL, you can run SQL*Loader by accessing the host with the exclamation point as shown below:

```
SQL> !sqlldr userid=system/manager
```

Fixed Format Data Files

The EMP and DEPARTMENT tables are created with the statements below. There are constraints placed on the EMP table for a Primary Key, a Salary Limit, and a Foreign Key as shown.

```
create table department
(Dept  Varchar2(4) primary key,
 DeptName varchar2(20));

insert into department values ('ACC','Accounting');
insert into department values ('MIS','Manage Info Sys');

create table emp
(ID number(3),
 Name varchar2(20,
 Salary number(6),
 Dept  varchar2(4),
 constraint emp_pk_const primary key (ID),
 constraint emp_sal_const check (salary < 100000),
 constraint emp_dept_fk_const foreign key (Dept)  references department);
```

The data file, named **emp_fixed.dat**, shown below contains data that are to be inserted into an empty EMP table.

```
60  Jim      50000  ACC
61  John     60000  MGT
67  James    50000  MIS
63  Jack     160000 MIS
64  Joan     70000  FIN
65  Jake     10000  MIS
66  Jim      40000  MIS
67  Frank    75000  ACC
68  Jack      xxxx  ACC
```

The data are in a fixed format. The ID's are contained in columns 1 through 2, the Names are located in columns 5 through 13, the Salaries are contained in columns 13 through 20, and the Dept's are in columns 21 through 23. Problems exist with the data in the highlighted rows. Employees 61 and 64 belong to departments that are not in the DEPARTMENT table which violates a foreign key constraint. Employee 63 earns more

than the $100,000 salary constraint will allow. Employee 67 is listed twice violating a primary key constraint. The data format for employee 68's salary is character instead of numeric. The records with errors should be caught by SQL*Loader.

The following conditions are placed on the SQL*Loader run:

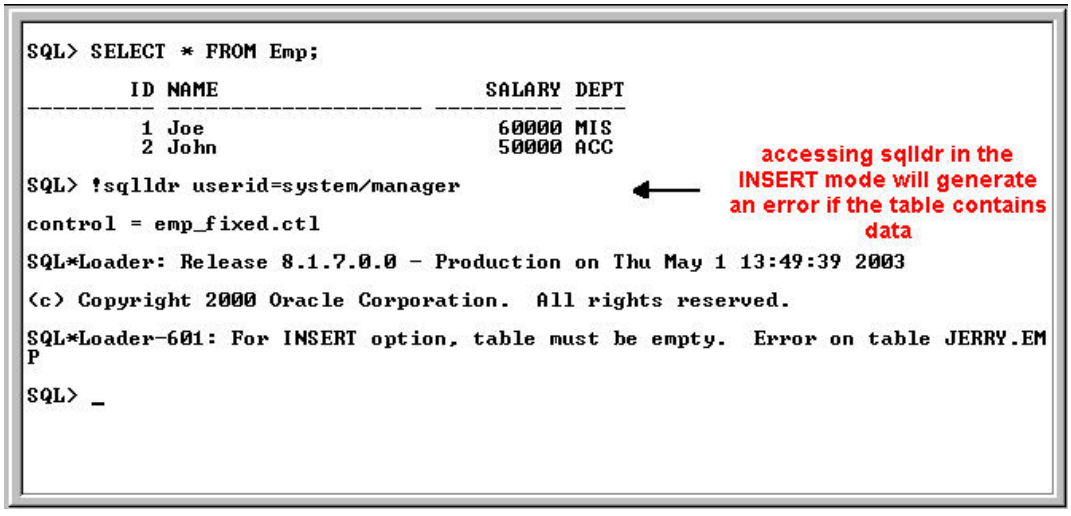
- The file should be loaded using the conventional path.
- The Table should be loaded using INSERT (i.e. the table must be empty).
- Any employee named *Jack* should not be inserted into the table.
- Rejected records should be listed in Bad and Discard files.

The contents of the Control File (named **emp_fixed.ctl**) is shown below. A log will be created automatically named **emp_fixed.log**.

```
Load data
  Infile 'emp_fixed.dat'
  Badfile 'emp_fixed.bad'
  Discardfile 'emp_fixed.dis'
INSERT
INTO TABLE jerry.emp
WHEN Name != 'Jack'
(Id          position(1:2)         char,
Name         position(5:12)        char,
Salary       position(13:20)       char,
Dept         position(21:23)       char)
```

Figure 3 shows the error that will be generated if you try to use the insert mode and data reside in the target table.

Figure 3
Using Insert Mode On A Populated Table



After removing the data from the EMP table, SQL*Loader is run again. As can been seen in Figure 4, the four rows "good" rows of data are inserted.

Figure 4
SQL*Loader Run

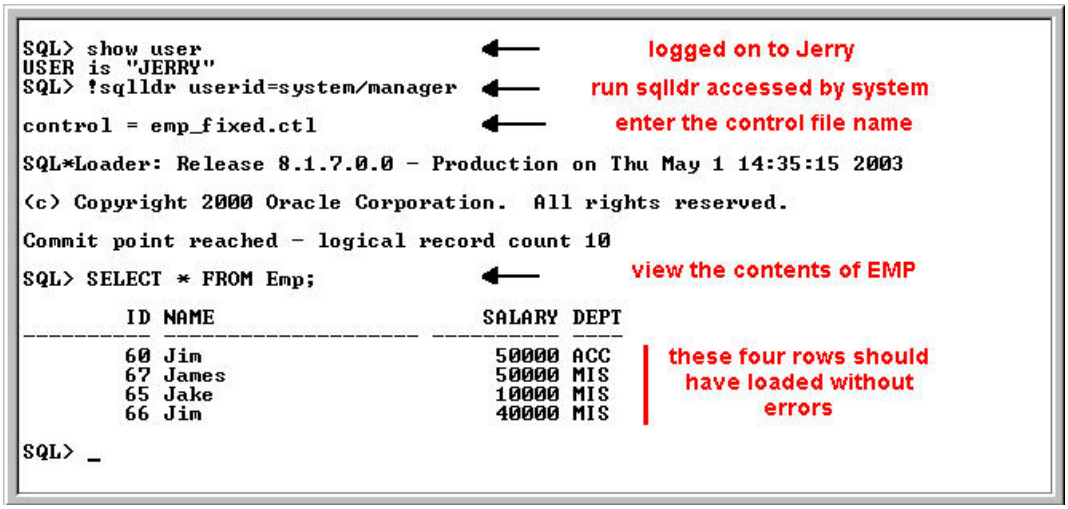


Figure 5 below shows the contents of the BAD and DISCARD files after the SQL*Loader run. Jack has been filtered out in the WHEN clause so his records are contained in the Discard File.

Figure 5
BAD And DISCARD File Contents

The contents of the **emp_fixed.log** file are shown below. The log provides a summary of the run including a description of the reason for rejecting records.

```
SQL*Loader: Release 8.1.7.0.0 - Production on Thu May 1 14:35:15 2003

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Control File:   emp_fixed.ctl
Data File:      emp_fixed.dat
  Bad File:     emp_fixed.bad
  Discard File: emp_fixed.dis
 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 65536 bytes
Continuation:    none specified
Path used:      Conventional

Table JERRY.EMP, loaded when NAME != 0X4a61636b(character 'Jack')
Insert option in effect for this table: INSERT

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
ID                                  1:2     2           CHARACTER
NAME                                5:13    9           CHARACTER
SALARY                             13:20    8           CHARACTER
DEPT                               21:23    3           CHARACTER

Record 4: Discarded - failed all WHEN clauses.
Record 9: Discarded - failed all WHEN clauses.
Record 10: Discarded - failed all WHEN clauses.
Record 2: Rejected - Error on table JERRY.EMP.
ORA-02291: integrity constraint (JERRY.EMP_DEPT_FK_CONST) violated - parent key not found

Record 5: Rejected - Error on table JERRY.EMP.
ORA-02291: integrity constraint (JERRY.EMP_DEPT_FK_CONST) violated - parent key not found

Record 8: Rejected - Error on table JERRY.EMP.
ORA-00001: unique constraint (JERRY.EMP_PK_CONST) violated


Table JERRY.EMP:
  4 Rows successfully loaded.
  3 Rows not loaded due to data errors.
  3 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  2048 bytes(64 rows)
Space allocated for memory besides bind array:      0 bytes

Total logical records skipped:          0
Total logical records read:            10
Total logical records rejected:         3
Total logical records discarded:        3

Run began on Thu May 01 14:35:15 2003
Run ended on Thu May 01 14:35:24 2003

Elapsed time was:     00:00:08.75
CPU time was:         00:00:00.06
```

Using The Direct Path

The Direct Path data load option will instruct SQL*Loader to disable integrity constraints on the table, load the data, and then attempt to enable the constraints. If data in the table

violate the constraints, they cannot be enforced and will remain in a disabled state. The **emp_fixed.ctl** control file is modified as shown below to use the direct path and preclude logging to the Redo Log Buffer.

```
OPTIONS (direct = true)
Unrecoverable
Load data
  Infile 'emp_fixed.dat'
  Badfile 'emp_fixed.bad'
  Discardfile 'emp_fixed.dis'
INSERT
INTO TABLE jerry.emp
WHEN Name != 'Jack'
(Id          position(1:4)         char,
Name         position(5:12)        char,
Salary       position(13:20)       char,
Dept         position(21:23)       char)
```

In Figure 6, the Delete command removes all of the data from EMP so the INSERT dataload mode can be used. SQL*Loader is run with the direct path option and unrecoverable in effect.

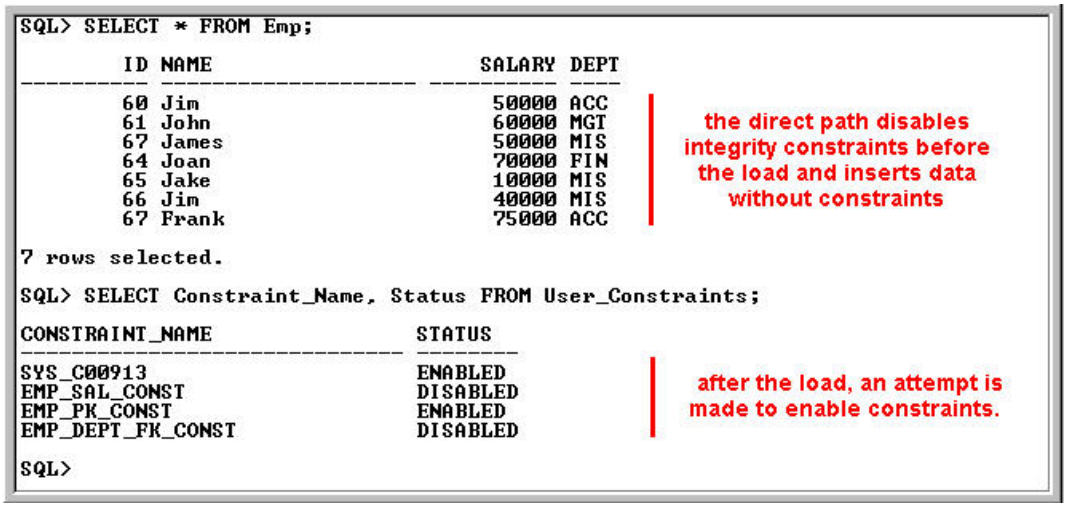
Figure 6
SQL*Loader Run Using The Direct Path

```
SQL> delete emp;

4 rows deleted.

Commit complete.
SQL> !sqlldr userid = system/manager

control = emp_fixed.ctl

SQL*Loader: Release 8.1.7.0.0 - Production on Thu May 1 15:35:22 2003

(c) Copyright 2000 Oracle Corporation.  All rights reserved.


Load completed - logical record count 10.

SQL>
```

sqlldr is run with the direct path option in effect

Figure 7 shows the data that reside in EMP after the SQL*Loader run. The data violate the integrity constraints that were initially placed on the table. A view from user_constraints shows that some constraints could not be validated and are left in a disabled state. Future Inserts and Updates on EMP will not be valided.

Figure 7
Results of SQL*Loader With The Direct Path Option



Variable Format Data Files

Sometimes data files arrive in a format in which the data are separated by some character such as a comma or white space. The data from a variable length file, called **emp_var.dat** are show below. The data items in the file are separated by commas and there is no comma at the end of a line. These data are to be added to the EMP table through SQL*Loader. The highlighted lines indicate records that will not be validated by the constraints on the EMP table.

```
70,Sue S,45000,MIS
71,Sandy A,55000,ACC
72,Jack,50000,MIS
72,Sally,90000,MIS
74,Sammy,175000,FIN
75,Sarah,80000,MIS
76,Randy,50000,MIS
78,Ralph,120000,ACC
```

The EMP table is created with all the constraints enabled. Two rows are inserted into the EMP table. The control file, called **emp_var.ctl** should be created according to the following conditions:

- The file should be loaded using the conventional path.
- The Table should be loaded using APPEND (i.e. the table need not be empty).
- Any employee earning a Salary of 45000 should not be inserted into the table.
- Rejected records should be listed in Bad and Discard files.

The contents of the control file are shown below. The 45000 value in the When clause is

in single quotes because it is listed as a Char type below. The clause “trailing nullcols” indicates that additional unused columns could exist in the input file. Trailing nullcols is not necessary in this example.

```
Load data
  infile 'emp_var.dat'
  badfile 'emp_var.bad'
  discardfile 'emp_var.dis'
APPEND
into table jerry.emp
when Salary != ‘48000’
Fields terminated by ','
trailing nullcols
(id,
name,
salary,
dept
terminated by whitespace)
```

Figure 8 shows that two rows exist in the EMP table before the data are loaded. The emp_var.ctl file from above is loaded into the program.

Figure 8
Loading Variable Format Data

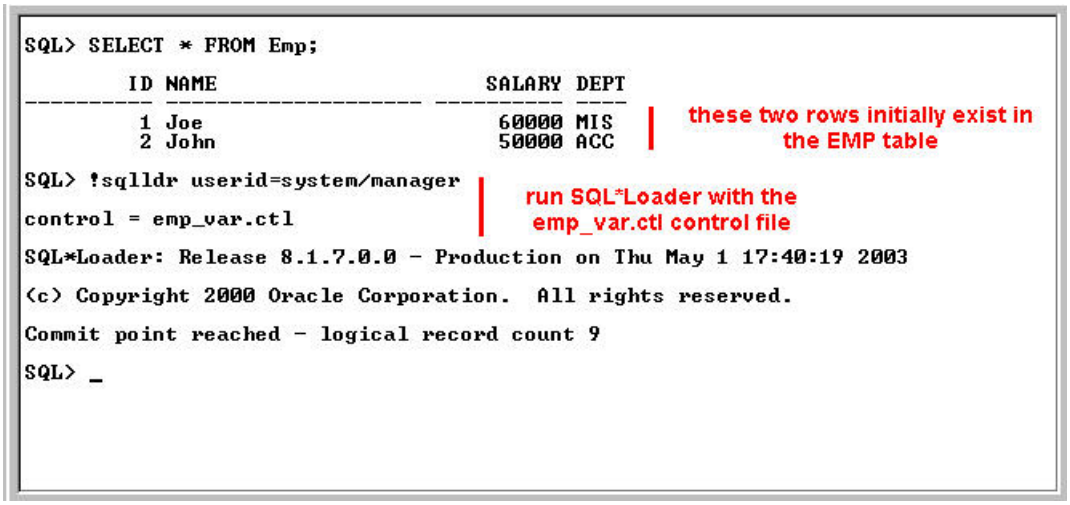
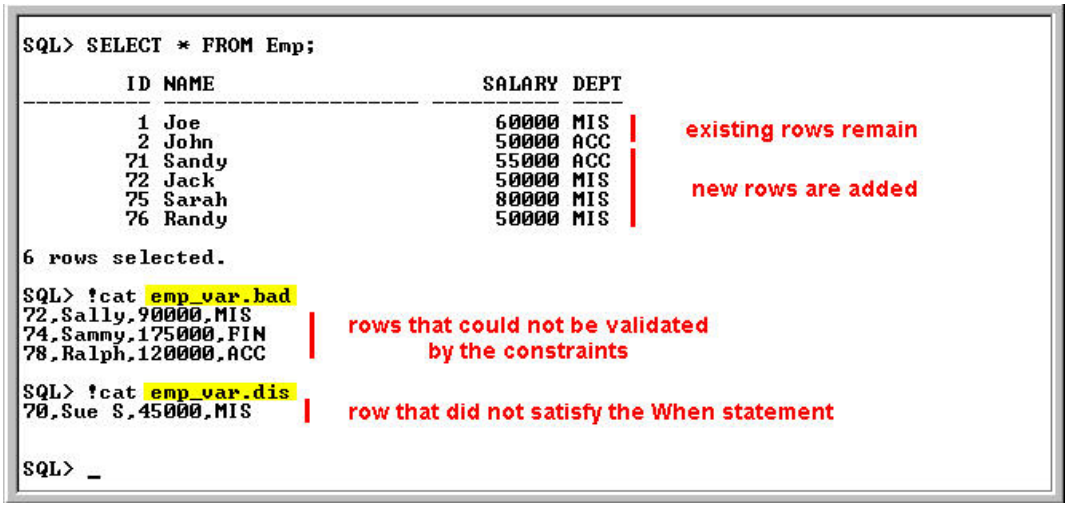


Figure 9 shows that the data are added to the EMP table because the Append mode was used. The Bad file contains the three records that violated table constraints and the Discard file contains the record that violated the When clause.

Figure 9
Table and File Contents After The Variable File Load



Loading Multiple Tables

SQL*Loader allows you construct parameters to load data into more than one table within a single control file. The INTO TABLE command can be written multiple times with different location specifications in the INFILE. The following data are contained in the **emp_fixed.dat** file. The first four columns identify rows to be inserted into the EMP table. The last two columns identify rows to be inserted into the DEPARTMENT table. Problems exist in the highlighted rows.

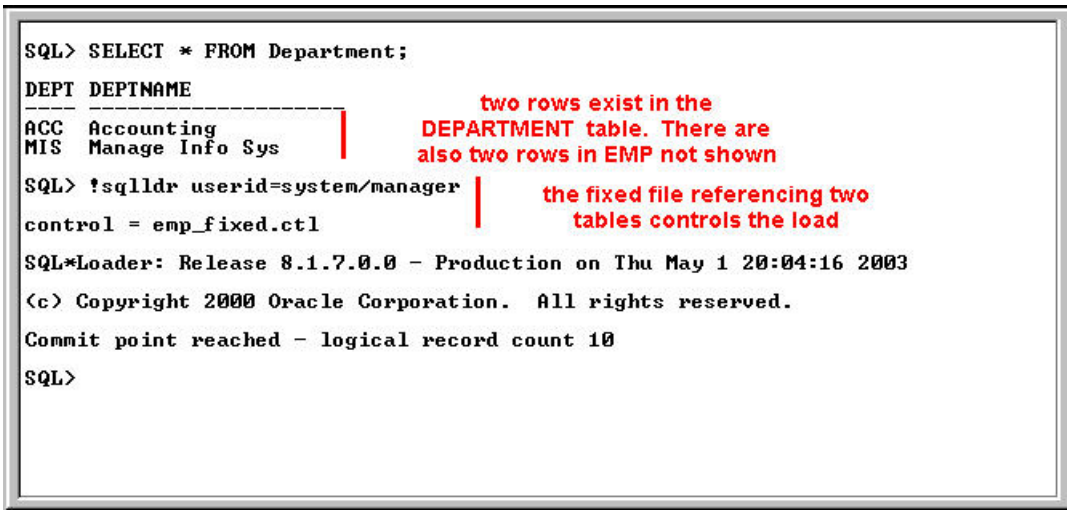
```
60  Jim      50000  ACC   MGT  Management
61  John     60000  MGT   MIS  Money System
67  James    50000  MIS
63  Jack     160000 MIS
64  Joan     70000  FIN
65  Jake     10000  MIS
66  Jim      40000  MIS
67  Frank    75000  ACC
68  Jack      xxxx  ACC
```

The control file, called emp_fixed.ctl, is shown below. The Infile identifies the file depicted immediately above. Only one Bad File and one Discard file are specified. The DISCARDMAX value limits the number of records that can be discarded before causing the load to terminate. DISCARDMAX can be used with any of the control files illustrated in this chapter. The data are to be loaded in the APPEND mode. The mode can be specified only once. You can't use the INSERT mode for the EMP table and the APPEND mode for the DEPARTMENT table. The INSERT INTO table_name clause is defined for the EMP table and then repeated for the DEPARTMENT table. Each INSERT INTO clause identifies where the records are located in the Infile.

```
load data
   infile 'emp_fixed.dat'
   badfile 'emp_fixed.bad'
   discardfile 'emp_fixed.dis'
append
into table jerry.emp
when Name != 'Jack'
(id       position (01:02)    char,
name      position (05:13)    char,
salary    position (13:20)    char,
dept      position (21:23)    char)
into table jerry.department
(dept     position (27:29)    char,
 deptname position (32:41)    char)
```

Figure 10 shows the contents of the DEPARTMENT table. The EMP table also contains two rows. The **emp_fixed.ctl** control file defined above controls the loading of the data file shown above.

Figure 10
Loading Data Into Two Table With One Control File



After the data are loaded, the EMP table and the DEPARTMENT table have the new records that are validated by the constraints. The contents of both the EMP and DEPARTMENT tables are shown in Figure 11. The records violating constraints are shown in the Bad file in Figure 12. The Discard file contains the records with employees named Jack which have been filtered by the When clause.

Figure 11
Tables After The Multiple Table Load

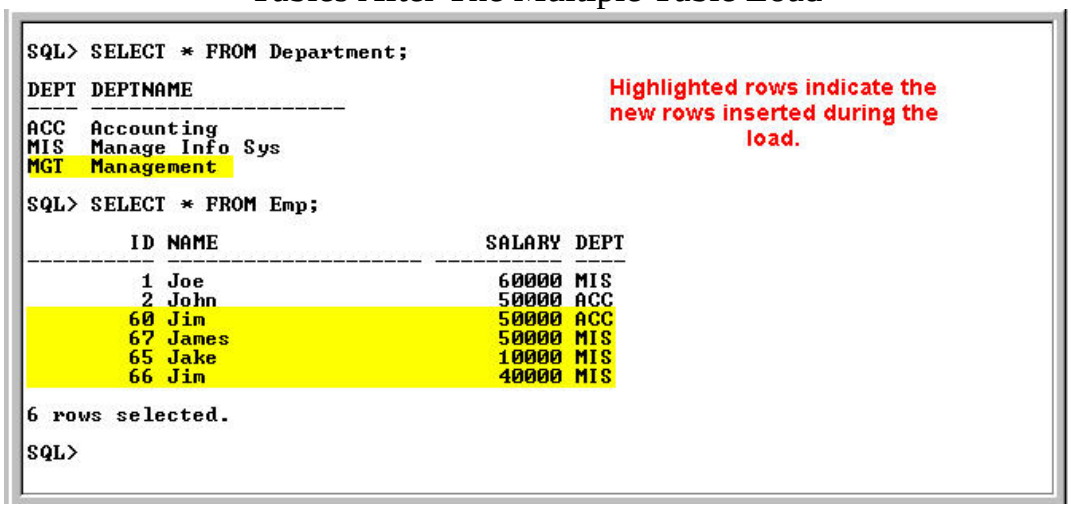


Figure 12
Rejected Record In A Multiple Table Load