ORACLE DATABASE 12c

An Oracle White Paper
August 2013

Express Mode Loading with SQL*Loader in Oracle Database 12*c*

Introduction

A common problem for users of SQL*Loader is generating the control file that SQL*Loader uses to load a data file. While SQL*Loader provides flexibility for loading different types of data files, that flexibility can make creating even simply control files for SQL*Loader more difficult than users expect. The more fields there are in the data file, the more room there is for errors when creating the control file.

Luckily, most data files are in comma-separated values (CSV) format. In Oracle Database 12*c*, SQL*Loader has a new feature called express mode that makes loading CSV files faster and easier. With express mode, there is no need to write a control file for most CSV files you load. Instead, you can load the CSV file with just a few parameters on the SQL*Loader command line. This white paper shows you how to get started with express mode.

Benefits of Using Express Mode

The main benefit of SQL*Loader express mode is the savings for time and effort that results from not needing to write and test a SQL*Loader control file. Instead, you specify a single SQL*Loader command with a few parameters and the load starts.

Another benefit of express mode is that it will try to use the fastest mechanism for loading data files: external tables using parallel inserts with the append hint. The append hint on an insert statement tells the database to use direct path for loading table data. Executing the insert in parallel means that multiple processes can load data from the data file at once, reducing the elapsed time for the load. If express mode finds that external tables cannot be used for a particular load operation, it will fall back to the next fastest mechanism, which is a direct path load in SQL*Loader.

Overview of CSV Files

Express mode for SQL*Loader loads data only from CSV files. CSV files have the following characteristics:

CSV files are text files where every record in the file contains data for one row in a table

The fields in each record are in the same order in every record.

Each record ends with a newline

On Unix-like operating systems, a newline is the linefeed character. On Windows, a newline is a carriage return and a linefeed character.

A character, usually a comma, terminates the value for each field in the record

Use of a comma is not a requirement. Other characters such as colon or vertical bar are also used commonly as terminators. For simplicity, we still call these files CSV files even if the terminator character is something other than a comma. The terminator after the last field in the record is optional.

CSV files can have another character that is used to enclose the field values

This character is usually the double quote character, but it can be another character. Enclosing a value allows you to have a newline or the field terminator character included as part of the value of the field.

Use of the enclosure character can be optional or mandatory

CSV files that use the enclosure character can require them around every field or let the use of enclosures be optional. If the use of enclosure is optional, then they are typically used only for field values that contain a newline or the terminator character.

A field can contain the enclosure character if "stutter" syntax is used

If a CSV file uses enclosure characters, and if a field value contains the enclosure character, then the enclosure character in the field value has to be duplicated. This is called "stutter syntax". It is a way to distinguish the occurrence of the enclosure character in the middle of a field from an enclosure character marking the end of the field.

Loading with Express Mode

In Oracle Database 12*c*, SQL*Loader has a new parameter, `TABLE`, that turns on express mode. The value of the `TABLE` parameter is the name of the table that SQL*Loader will load. If `TABLE` is the only parameter specified, then SQL* loader will do the following:

1. Look for a data file in the current directory with the same name as the table being loaded and with an extension of ".dat". The case of the name of the data file is the same as the case of the table name specified for the `TABLE` parameter
2. Assume the order of the fields in the data file matches the order of the columns in the table
3. The fields are terminated by commas, but there is no enclosure character

Here is a simple example of a table, a data file, and the SQL*Loader command that will load the table. The table is created with the following command:

```
CREATE TABLE EMP
(EMPNO    number(4) not null,
 ENAME    varchar2(10),
 HIREDATE date,
 DEPTNO   number(2));
```

The data file used to load the table is derived from the table name, `emp`, and is `emp.dat`. The content of the data file in this example is shown here:

```
7782,Clark,09-Jun-81,10
7839,King,17-Nov-81,12
```

Note that the order of the fields in the data file matches the order of the columns in the table. The following SQL*Loader command will load the table from the data file.

```
sqlldr userid=scott table=emp
```

Note that no control file is needed for this example. After executing the SQL*Loader command, a SELECT from the table will show the following:

```
     EMPNO ENAME      HIREDATE       DEPTNO
---------- ---------- --------- ----------
      7782 Clark      09-JUN-81          10
      7839 King       17-NOV-81          10
```

Customizing SQL*Loader Express Mode

While the SQL*Loader command in the previous example was simple, it made some assumptions that may not always be true. For example, CSV files can use a character other than a comma to terminate fields or there might be multiple data files to load, none of which use the name of the table in the file names. For these reasons, SQL*Loader Express Mode has other command line parameters that allow you to modify the load.

Changing the Field Terminator

The `TERMINATED_BY` parameter for the SQL*Loader command line specifies the character that is used to terminate the fields. If the data file in the example used vertical bars for separators rather than commas, it would look like this:

```
7782|Clark|09-Jun-81|10
7839|King|17-Nov-81|12
```

The command used to load this data file is:

```
sqlldr userid=scott table=emp terminated_by='|'
```

Specifying the Order of the Fields in the Data File

When SQL*Loader Express Mode reads fields the record in the data file, it uses the order of the columns in the table as the order of the fields in the data file. However, it is possible that the order of the fields in the data file is different than the order of the columns in the table. For those data files, use the new `FIELD_NAMES` parameter. This

parameter tells SQL Loader that the first record in the data file specifies the order of the fields in the data file.

This example shows a data file `emp.dat` used to load table `EMP` where the field order does not match the order of the columns in the table.

```
deptno,empno,ename,hiredate
10,7782,Clark,09-Jun-81
12,7839,King,17-Nov-81
```

The command used to load this data file is:

```
sqlldr userid=scott table=emp field_names=first
```

The `FIELD_NAMES` parameter has other options to customize loading of multiple data files. In this example, `FIELD_NAMES=FIRST` tells SQL*Loader that only the first data file has the field name list as its first record. If the load uses multiple data files, none of the other data files will have a field name list. Specifying `FIELD_NAMES=ALL` tells SQL*Loader that all data files have the list of field names as its first record.

Using a Different Data File

The default name for the data file is the name of the table with a `.dat` extension. To load a file with a different name or to load multiple files, use the `DATA` parameter. The `DATA` parameter accepts a list of values, so you can specify multiple files as part of the load.

Starting in Oracle Database 12*c*, SQL*Loader also supports wildcard characters, "*" and "?" in file names. The "*" wild card character matches one or more characters in a file name. The "?" wild card character matches just one character. This greatly simplifies the command string when loading dozens or hundreds of files with similar names.

Assume we have two files to load into table `emp`. The first file is `jan_2012_emp.dat`:

```
7782,Clark,09-Jan-12,10
7839,King,17-Jan-12,12
```

The second file is `feb_2012_emp.dat`:

```
8109,Baby,12-Feb-12,10
8299,Lee,24-Feb-12,12
```

You could load these files with either of the two following commands for SQL*Loader:

```
sqlldr userid=scott table=emp data=jan_2012_emp.dat,feb_2012_emp.dat

sqlldr userid=scott table=emp data=*_2012_emp.dat
```

The first command lists the files to be loaded. The second command uses wild cards in the filenames for the data parameter.

Using an Enclosure Character

A data file may need to use an enclosure character around fields because the fields might contain the terminator character or newline characters as part of the value of the field. Consider the following table definition:

```
create table part_info
(part_number       varchar2(10),
 part_description  varchar2(40)
);
```

File `part_info.dat` contains the following:

```
10-1002,"size 12 widget, red"
10-1003,"size 12 widget, blue"
```

Table `PART_INFO` could be loaded with the following command:

```
sqlldr userid=scott table=part_info optionally_enclosed_by=\'\"\'
```

After the load, the table would have the following rows.

```
PART_NUMBE PART_DESCRIPTION
---------- ----------------------------------------
10-1002    size 12 widget, red
10-1003    size 12 widget, blue
```

This example used `OPTIONALLY_ENCLOSED_BY` because only some of the field values were enclosed by quotes. Use the `ENCLOSED_BY` parameter if every field is enclosed.

Using an Enclosure Character with for Fields Containing NEWLINE

The enclosure character can be used to enclose fields that also contain newlines. For example, here is a table definition with a large text field.

```
CREATE TABLE EMP_RESUMES
(EMPNO      number(4) not null,
 EMP_RESUME varchar2(1000));
```

Here is the data file that needs to be loaded into the table.

```
7782,"line 1 of resume for 7782
line 2 of resume
```

```
last line of resume"
7839,"line 1 of resume for 7839
line 2 of resume
last line of resume"
```

Note that the EMP_RESUME field is enclosed by double quotes and has newlines embedded in the data. If a data file does contain such a field, then the SQL*Loader command needs to specify the CSV parameter with a value of WITH_EMBEDDED. SQL Loader needs to do extra processing to handle files that contain newline characters inside of fields. This extra processing can slow the load significantly for data files that do not require it. So, rather than make all loads pay the price for this processing, SQL*Loader requires the user to specify CSV=WITH_EMBEDDED to do the special processing when loading data files that need it.

The SQL Loader command to load table EMP_RESUMES with the data file shown above is:

```
sqlldr userid=scott table=emp_resumes csv=with_embedded
```

Selecting from the table after the load shows the data from the file:

```
SQL> select * from emp_resumes;
     EMPNO
----------
EMP_RESUME
--------------------------------------------------------------------------------
      7782
line 1 of resume for 7782
line 2 of resume
last line of resume
      7839
line 1 of resume for 7839
line 2 of resume
last line of resume
```

Other Customizations

SQL*Loader Express Mode supports some additional parameters to customize a load. These are listed below. The documentation for SQL*Loader express mode has more information about these options.

- BAD – the file where SQL*Loader writes records that could not be loaded.
- CHARACTERSET – the name of the character set used to encode the data files.
- DATE_FORMAT – the format string to use when interpreting dates in the data file.
- DEGREE_OF_PARALLELISM – the degree of parallelism to use when loading with external tables.

- `DIRECT` – force the load to use direct path or conventional path load instead of external tables.
- `EXTERNAL_TABLE` – force the load to use external tables to load the data file, or tell SQL*Loader to write the SQL statements used to load the external tables to the log file without executing those statements.
- `LOAD` – the number of records to load. The default is to load all records.
- `NULLIF` – the value for fields in the record that will cause SQL*Loader to insert a `NULL` for that field.
- `SILENT` – control what messages are written to the SQL Loader log file.
- `TIMESTAMP_FORMAT` – the format string to use when interpreting timestamp fields in the data file.
- `TRIM` – how white space should be trimmed from the beginning and end of a field.

SQL*Loader Log File for Express Mode

The log file for SQL*Loader Express Mode contains the same information as other SQL*Loader operations, but it also has additional information. SQL*Loader Express Mode writes the contents of a SQL*Loader control file that would perform the same load as if express mode were not being used.

If the customization options of express mode are not sufficient for a load, then you can use the control file information in the log files as a starting point for writing and customizing a control file to load the data. For example, the `DATE_FORMAT` parameter does not help if there are different data formats used by different date fields in the data file. In that case, you can

- Use SQL*Loader express mode to write the control file information to the log file,
- Create a new control file with that information, and then
- Modify the data format strings to the correct values for the date fields.

This makes SQL*Loader express mode a good way to learn how to write more complicated SQL*Loader control files.

Conclusion

Express mode in SQL*Loader greatly simplifies the work needed to load most data files. Use express mode for loading CSV files, the most common file format. Express mode tries to use the fastest mechanism possible for loading the data, and has optional parameters that allow users to customize the load. Getting your load started and completed is faster with express mode.

Express Mode Loading with SQL*Loader in Oracle Database 12*c*
June 2013
Author: Jim Stenoish
Contributing Authors: Ray Pfau, Roy Swonger

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together