

DB2 Load and Oracle SQL*Loader

A technical overview

Skill Level: Introductory

[Burt L. Vialpando \(burt.vialpando@us.ibm.com\)](mailto:burt.vialpando@us.ibm.com)

Certified Consulting I/T Specialist
IBM

[Vikram S. Khatri \(vikram@zinox.com\)](mailto:vikram@zinox.com)

Certified Consulting I/T Specialist
IBM

14 Aug 2008

The Oracle SQL*Loader uses a control file to load data and so does the DB2 LOAD utility. However, the structure of each of these control files is quite different, and many Oracle and DB2 DBAs would like to know how these compare. In this article, get a comparison of these two utilities and learn how to use a Perl tool to transform Oracle SQL*Loader scripts to DB2 LOAD scripts. Since space is always at premium in large data warehouses, this article also explains the way to modify data files in-flight for a DB2 LOAD.

DB2 LOAD in comparison to Oracle SQL*Loader

DB2 LOAD versus DB2 IMPORT which "path" do you use?

Note:

The DB2 IMPORT utility is not tied to reading a proprietary format as is the Oracle IMPORT utility, so these two utilities should not be compared to each other.

Actually, DB2 has two different utilities for putting data from an external source into a DB2 table: LOAD and IMPORT. LOAD puts data in at the page level, bypasses trigger firing and logging, and puts off constraint checking and index building until after the

data has been put into the DB2 table. `IMPORT` on the other hand basically performs `INSERTS` and obeys trigger firing, performs logging, and does constraint checking and index building as it puts data into the table. There are many other differences between the options for these two utilities, however, it is out of the scope of this article.

The Oracle `SQL*Loader` utility on the other hand has two main modes, or paths, of operation: direct path and conventional path. Oracle DBAs use this one utility, but specify the "path", which is used for similar reasons as the two DB2 utilities.

`SQL*Loader` "direct path" mode is similar in functionality to DB2 `LOAD`.

`SQL*Loader` "conventional path" mode is similar in functionality to DB2 `IMPORT`.

Note:

It is not practical to try to exhaustively compare each and every DB2 `LOAD`, and for that matter `IMPORT`, feature (keyword with its options) with each and every Oracle `SQL*Loader` feature (keyword with its options.)

So this article covers the major features that are typically encountered by the Oracle DBA who wants to quickly migrate their `SQL*Loader` scripts to DB2 `LOAD` scripts.

As DB2 migration experts, over the years we have experienced that most Oracle DBAs use the conventional path mode of `SQL*Loader` most of the time, and most of their expertise and scripts are conventional path oriented. In fact, some Oracle DBAs never use the direct path mode of the `SQL*Loader` at all. However, when they begin to learn DB2, they often opt for using the DB2 `LOAD` utility (probably because of its name) and struggle using it because it has many of the characteristics of `SQL*Loader` direct path mode that they were not using before. So, to clarify, even though most Oracle DBAs use conventional path load options most of the time, this article demonstrate converting all `SQL*Loader` scripts, regardless of the path, to DB2 `LOAD` utility scripts. We think this gets the maximum performance out of DB2 right away. If for any reason a DBA wants to change these scripts to use `IMPORT` instead of `LOAD`, he can do so later if the situation warrants it.

The `SQL*Loader` command line invoking `SQLLDR`

The Oracle `SQL*Loader` utility is invoked with the binary `SQLLDR` and uses a command line syntax similar to that of the DB2 `LOAD`. The command line can contain many keywords that will indicate to the `SQL*Loader` utility where messages will go, where discarded records will go, and so on.

The `SQLLDR` command line also indicates the name of the "control file", which usually has the extension of `.CTL`. This control file can also indicate to the `SQL*Loader` utility where messages will go, where discarded records will go, and so on. A `SQLLDR` command line reference to the same keyword will override the control file reference of the same functionality, so you have to pay close attention to

both the SQLLDR command line and the control file in order to know how the SQL*Loader session will actually work. This was probably designed to provide flexibility and power for the SQL*Loader utility, but for those of us migrating these scripts to DB2, they can be confusing if some of the same keywords are being used in both places and the use of these are not consistent from script to script.

The SQL*Loader control file is the place where the heart and detail of the load functionality is described, so when we describe a comparison of SQL*Loader and DB2 LOAD, we are comparing for the most part the SQLLDR control file and the DB2 LOAD command line. First though, let's cover all the options of the SQLLDR command line and compare them to the DB2 LOAD command line to see similarities with these. Then we'll cover the control file and its keywords, and options to see how these also compare to the DB2 LOAD command line.

Table 1. DB2 LOAD versus Oracle SQL*Loader Direct Path

Feature	DB2 LOAD	SQL*Loader (Direct Path)
Direct Path	Use DB2 LOAD Utility	Use SQL*Loader Direct Path -- many options of conventional path not available.
Generate statistics after load	Yes, if data is replaced and not if appended	No
Recoverable	Yes with COPY YES option	No in Direct Path
Defaults (value)	Available with DB2 LOAD	Not available with direct path
Many input file for LOAD	Yes	One file at a time with one SQLLDR
Exception data	To an exception table and/or DUMP file	To an exception table
LOAD from CURSOR	Yes	No
LOAD from PIPE	Yes	Yes
BLOBS/CLOBS	Yes	Yes
XML Document	Yes	Yes
Compression allowed while loading	Yes	Yes
Online LOAD	Yes -- Access to table is available	No in direct path
Can you modify data while loading?	Yes -- through a user exit	Yes -- Conventional path (SQL Strings) No -- Direct Path
Populate constant values in columns	No	Yes using CONSTANT keyword
LOAD in multiple database partitions	Yes	N/A
How to invoke?	It is a DB2 command callable	It is a standalone utility callable

	from a SQL script or through an application using an API	from command line and through an API in an application
How to monitor LOAD status?	From another connection, run LOAD QUERY or LIST UTILITIES command	See through LOG file
Parallelism	Highly optimized to use multiple CPUs using multiple processes and threads	Can be parallelized using multi-threading

Listing 1. Oracle SQLLDR command line syntax example

```
SQLLDR CONTROL=sample.ctl DATA=sample.dat LOG=sample.log BAD=sample.bad
DISCARD=sample.dsc
      USERID=scott/tiger ERRORS=999 LOAD=2000 DISCARDMAX=5
```

Table 2. Oracle SQLLDR command line keyword comparison to DB2 LOAD keywords

Oracle SQLLDR keyword	Oracle SQLLDR keyword description	DB2 LOAD keyword	DB2 LOAD keyword description
CONTROL =filename.ctl	File with detailed LOAD command options.	N/A	DB2 LOAD command options are not separately called from a control file. The DB2 LOAD command line contains all the keywords in one invocation.
DIRECT =true	Invoke DIRECT PATH mode of the Oracle SQL*Loader utility.	LOAD	DB2 LOAD utility itself is the near equivalent to the Oracle Direct Path mode of the SQL*Loader utility.
DIRECT =false	If not used or valued at "false" then the CONVENTIONAL PATH mode of the Oracle SQL*Loader utility is invoked.	IMPORT	DB2 IMPORT is the near equivalent to the Oracle Conventional Path mode of the SQL*Loader utility.
BAD =filename.bad	Where to store rejected records.	MODIFIED BY DUMPFIL E=filename	DB2 LOAD modifier used to determine where rejected records are written.

DATA=filename.dat	Input data source file.	FROM sourcename	DB2 LOAD sourcename can be a file, pipe, device or cursor.
DISCARD=filename.dsc	Exception records not loaded for a variety of reasons.	FOR EXCEPTION tablename	DB2 LOAD puts records that violate unique index rules (exceptions) into a previously created table.
DISCARDMAX=number	Defines maximum amount of discard records before SQLLDL terminates.	WARNINGCOUNT=number	DB2 LOAD terminates on this many warnings. Discards are just one type of warning.
ERRORS=number	Defines maximum amount of errors before SQLLDL terminates.	NOROWWARNINGS	Modifier NOROWWARNINGS can turn off row warnings, but still leave a warning for exception records.
LOAD=number	Number of records to be loaded. (ALL=default)	ROWCOUNT number	Specifies number of records to be loaded. When omitted, the default is all.
MULTITHREADING=true	Allows for stream building on the client side and stream loading on the server side.	CPU_PARALLELISM number DISK_PARALLELISM number FETCH_PARALLELISM yes	DB2 LOAD determines these for itself using autonomics to control the number or threads spawned for parsing, converting, formatting and writing records in file, device, pipe and cursor loads. These can be specified if desired with these three keywords.
ROWS=number	Rows per data save.	SAVECOUNT number	DB2 LOAD uses consistency points for recoverability of the load operation.
LOG=logfile	LOG stores output from load operations.	MESSAGES messagefile	DB2 puts messages to this message file. It suppresses messages if you do not specify a message file.

SILENT= options	SILENT=options can suppress message output for various portions of that operation.	NOROWWARNINGS	Modifier NOROWWARNINGS turns off portions of the load operation message output.
SKIP= number	Start load after n records. Usually used to restart a load operation that committed a partial load but did not complete. Note: If using this feature, SQL*Loader requires the operators to determine this for themselves, and picking the wrong number could mean lost or duplicate data.	RESTART (REPLACE, INSERT, TERMINATE)	One of the modes that DB2 LOAD uses to pick up where it left off after the last consistency point it took before failure. DB2 LOAD determines for itself where to pick up and does not require operators to figure it out. The other modes DB2 LOAD can execute under are REPLACE, INSERT and TERMINATE, but these do not correlate to the SKIP keyword in any way.
SKIP_INDEX_MAINTENANCE	Skips index maintenance and marks indexes as unusable.	INDEXING MODE DEFERRED	DB2 LOAD can defer index refresh to occur later during access to the data or during database activation.
SKIP_UNUSABLE_INDEXES	Skips index maintenance on any indexes already marked unusable.	INDEXING MODE REBUILD, INCREMENTAL, AUTOSELECT	DB2 LOAD can also specify INDEXING MODE REBUILD, INCREMENTAL or AUTOSELECT which can specify how the LOAD will perform index maintenance.
READSIZE= number	Size of external data file read before a commit is required.	DATA BUFFER number	DB2 LOAD uses this many 4K pages to transfer data within the utility which is normally determined intelligently through autonomics, but can be specified if desired with this keyword.

USERID/PASSWORD	Userid that connects to the database.	CONNECT TO...	DB2 uses a connect command prior to any subsequent LOAD commands.
------------------------	---------------------------------------	----------------------	---

The SQL*Loader control file – the heart of SQL*Loader utility

Although the Oracle SQL*Loader command line can contain many keywords to control how the utility functions, our experience is that most of these keywords are usually indicated through the control file instead of the command line. Let's explore an example of a typical SQL*Loader control file converted to a DB2 LOAD command line.

Table 3. DB2 LOAD scripts converted from typical Oracle SQL*Loader control files

DB2 LOAD command file example INSERT fixed data	Oracle SQL*Loader control file example INSERT fixed data
<pre>(1) LOAD (2) FROM 'INPUT_FILE1.DAT' (3) OF ASC (4) MODIFIED BY DUMPFILE='INPUT_FILE1.B (5) METHOD L (1 5, 6 15, 16 20) (6) INSERT INTO PROD.TB_TABLE1 (7) (COL1, COL2, COL3) (8) FOR EXCEPTION PROD.TB_TABLE1_DSC ;</pre>	<pre>(1) LOAD DATA (2) INFILE 'INPUT_FILE1.DAT' (4) BADFILE 'INPUT_FILE1.BAD' (8) DISCARDFILE 'INPUT_FILE1.DSC' (6) APPEND INTO TABLE PROD.TB_TABLE1 (5)(7)(COL1 POSITION(01:05), COL2 POSITION(06:15), COL3 POSITION(16:20)) ;</pre>
DB2 LOAD command file example REPLACE variable data	Oracle SQL*Loader control file example REPLACE variable data
<pre>(1) LOAD (2) FROM 'INPUT_FILE2.DAT' (3) OF DEL (4) MODIFIED BY DUMPFILE='INPUT_FILE2.B (3) COLDEL (3) CHARDEL" (5) METHOD P (1, 2, 3) (6) REPLACE INTO PROD.TB_TABLE2 (7) (COL1, COL2, COL3) (8) FOR EXCEPTION PROD.TB_TABLE2_DSC ;</pre>	<pre>(1) LOAD DATA (2) INFILE 'INPUT_FILE2.DAT' (4) BADFILE 'INPUT_FILE2.BAD' (8) DISCARDFILE 'INPUT_FILE2.DSC' (6) REPLACE INTO TABLE PROD.TB_TABLE2 (3) FIELDS TERMINATED BY ' ' (3) OPTIONALLY ENCLOSED BY '"' (5)(7)(COL1, COL2, COL3) ;</pre>

The following descriptions of the load script sections compare to the above

examples:

- **(1) LOAD**
This invokes the `LOAD` utility in DB2, or you can use `IMPORT` to invoke that utility.

In Oracle, `LOAD DATA` is used to invoke the `SQL*Loader` utility. To specify a direct path load, you must say `DIRECT=true`. The default is `DIRECT=false` and is thus not shown in the example.

- **(2) FROM [inputfile_name]**
This is the file name that contains the data to be loaded. DB2 `LOAD` can also load data from a pipe, cursor, or device.

Oracle also names the input file or pipe, and is able to name data with a `BEGIN ...END` clause that functions as inline data through control file.

- **(3) OF ASC / DEL**
For DB2 `LOAD`, `ASC` means un-delimited ASCII data and that the data is positional. `DEL` means delimited ASCII and the data can be variable in length for each row. There are a number of modifiers for delimited data, but the two key ones are `COLDEL`, which determines how columns are delimited from column to column, the default is a comma, and `CHARDEL`, which determines how character data is delimited; the default is double quote.

Oracle has `FIX` (the default and thus not shown in the example) or `VAR`, but these keywords are rarely used. It is usually the use of other keywords and the insert column references that determine whether or not the data is fixed or variable. For example, keywords `FIELDS TERMINATED BY` and `FIELDS ENCLOSED BY` to function similarly as `COLDEL` and `CHARDEL` in DB2 `LOAD`, which are required for variable, delimited data.

- **(4) MODIFIED BY DUMPFIL=[dumpfile_name]**
DB2 puts rejected records to this file.

Oracle uses the `BADFILE` keyword to accomplish the same thing.

- **(5) METHOD P (1,2,3)**
DB2 `LOAD` has three methods:

1. `METHOD L` is for `ASC` data only, and this method tells the start

and end of each column. It looks like this: METHOD L (start1 end1, start2 end2....)

2. METHOD N is for IXF or cursor data, and this names the columns in the source table that are being loaded. It looks like this:
METHOD N (col1, col2, col4...)
3. METHOD P is for DEL, IXF, or cursor data, and this gives the positional number of the columns from the source data that are being loaded. It looks like this: METHOD P (1, 2, 4...)

SQL*Loader can combine using a column name as well as position of the fields in one line, as shown in the example.

- **(6) INSERT / REPLACE INTO PROD.TABLE**

DB2 LOAD has four options here. The two that correspond to Oracle SQL*Loader are INSERT and REPLACE. The other two DB2 LOAD options you can use are RESTART and TERMINATE. These are used when a DB2 LOAD does not complete for any reason.

SQL*Loader also has INSERT, but this is only used for an empty table, and since APPEND works like INSERT on an empty table, few Oracle DBAs use it. SQL*Loader REPLACE works the same as DB2 LOAD REPLACE.

- **(7) (COL1, COL2, COL3) Insert Column List**

DB2 LOAD will use this list of columns to place the data into. If you omit the column list, DB2 LOAD will attempt to load the data from the first through the last column with the data as it is read and parsed from the first field through the last field.

SQL*Loader can combine using a column name as well as position of the fields in one line, as shown in the example. For variable length data, position is not given but instead is determined by delimiters.

- **(8) FOR EXCEPTION [table_name]**

DB2 LOAD writes records that violate unique index rules (exceptions) to this previously created table.

SQL*Loader uses DISCARDFILE for the same thing, only it is an OS file and not a DB2 table.

Table 4. Oracle SQLLDR control file keyword comparison to DB2 LOAD keywords

Oracle SQLLDR	Oracle SQLLDR	DB2 LOAD keyword	DB2 LOAD keyword
---------------	---------------	------------------	------------------

keyword	keyword description		description
DIRECT=true	Invoke DIRECT PATH mode of the Oracle SQL*Loader utility.	LOAD	DB2 LOAD utility itself is the near equivalent to the Oracle Direct Path mode of the SQL*Loader utility.
DIRECT=false	If not used or valued at "false", the CONVENTIONAL PATH mode of the Oracle SQL*Loader utility is invoked.	IMPORT	DB2 IMPORT is the near equivalent to the Oracle Conventional Path mode of the SQL*Loader utility.
ERRORS=number	Defines maximum amount of errors before SQLLDR terminates.	NOROWWARNINGS	Modifier NOROWWARNINGS can turn off row warnings, but still leave a warning for exception records.
LOAD=number	Number of records to be loaded. (ALL=default)	ROWCOUNT number	Specifies number of records to be loaded. When omitted, the default is all.
MULTITHREADING=true	Allows for stream building on the client side and stream loading on the server side.	CPU_PARALLELISM number DISK_PARALLELISM number FETCH_PARALLELISM yes	DB2 LOAD determines these for itself using autonomics to control the number or threads spawned for parsing, converting, formatting, and writing records in file, device, pipe, and cursor loads. These can be specified if desired with these three keywords.
READSIZE=n	Size of external data file read before a commit is required.	DATA BUFFER number	DB2 LOAD uses this many 4K pages to transfer data within the utility, which is normally determined intelligently through autonomics, but can be specified if desired with this keyword.
ROWS=number	Rows per data save.	SAVECOUNT number	DB2 LOAD uses

			consistency points for recoverability of the load operation.
SILENT= options	SILENT=options can suppress message output for various portions of that operation.	NOROWWARNINGS	Modifier NOROWWARNINGS turns off portions of the load operation message output.
SKIP= number	Start load after n records. Usually used to restart a load operation that committed a partial load but did not complete. Note: If using this feature, SQL*Loader requires the operators to determine this for themselves, and picking the wrong number could mean lost or duplicate data.	RESTART (REPLACE, INSERT, TERMINATE)	One of the modes that DB2 LOAD uses to pick up where it left off after the last consistency point it took before failure. DB2 LOAD determines for itself where to pick up and does not require operators to figure it out. The other modes DB2 LOAD can execute under are REPLACE, INSERT, and TERMINATE, but these do not correlate to the SKIP keyword in any way.
SKIP_INDEX_MAINTENANCE	Skips index maintenance and marks indexes as unusable.	INDEXING MODE DEFERRED	DB2 LOAD can defer index refresh to occur later during access to the data or during database activation.
SKIP_UNUSABLE_INDEX	Skips index maintenance on any indexes already marked unusable.	INDEXING MODE REBUILD, INCREMENTAL, AUTOSELECT	DB2 LOAD can also specify INDEXING MODE REBUILD, INCREMENTAL, or AUTOSELECT, which can specify how the LOAD will perform index maintenance.
NOLOGGING	This option allows logging to be bypassed, but makes the table unrecoverable with a roll forward operation.	NONRECOVERABLE	With this option, table spaces are not put in backup pending state following the load operation, and a copy of the loaded data does

			not have to be made during the load operation.
CONTINUE_LOAD DATA	Starts a terminated load by automatically finding the proper place to continue. (Direct path only.)	RESTART	DB2 LOAD uses the last consistency point it took before failure to pick up where it left off.
LOAD DATA	Invoke SQLLDR binary to load data with any mode (or path).	1. LOAD 2. IMPORT	DB2 LOAD utility itself is the near equivalent to the Oracle Direct Path mode of the SQL*Loader utility. DB2 IMPORT is the near equivalent to the Oracle Conventional Path mode of the SQL*Loader utility.
INFILE filename	Input data source file.	FROM sourcename	DB2 LOAD sourcename can be a file, pipe, device, or cursor.
RECSIZE n	Size of fixed input record.	MODIFIED BY RECLEN=x	Size of fixed input record.
BADFILE filename	Where to store rejected records.	MODIFIED BY DUMPFILE=filename	DB2 LOAD modifier used to determine where rejected records are written.
DISCARDFILE filename	Exception records not loaded for a variety of reasons.	FOR EXCEPTION tablename	DB2 LOAD puts records that violate unique index rules (exceptions) into a previously created table.
DISCARD	Exception records not loaded for a variety of reasons.	FOR EXCEPTION filename	DB2 LOAD puts records that violate unique index rules (exceptions) into a previously created table.
DISCARDMAX	Defines maximum amount of discard	1. WARNINGCOUNT=number	DB2 LOAD terminates this many warnings.

	records before SQLLDR terminates.	2. NOROWWARNINGS	Discards are just one type of warning. Modifier NOROWWARNINGS can turn off row warnings, but still leave a warning for exception records.
1. VAR n 2. FIX n	Expects data to be variable or fixed in formatting. If VAR is used, then n is the number of bytes at the beginning of the row that declares how long each row is.	1. OF DEL 2. OF ASC	DB2 variable data is delimited, and fixed data is ASCII.
1. INSERT or APPEND 2. REPLACE	INSERT and APPEND are similar in that they add to data already existing, except INSERT expects the table to be empty. REPLACE truncates any data already in the table before adding the new data.	1. INSERT 2. REPLACE	DB2 LOAD INSERT adds to the data in the table, even an empty one. REPLACE truncates any data already in the table before adding the new data.
INTO TABLE tablename	Table the data is going into.	INTO TABLE tablename	Table the data is going into.
TERMINATED BY string	Data is read until the first occurrence of this string.	MODIFIED BY COLDELx	All input data is delimited from column to column by this character. (Default is a comma.)
ENCLOSED BY string	Data is optionally enclosed by this string; usually this is for character data.	MODIFIED BY CHARDELx	Input character data is enclosed with this character. (Default is double quote.)
LOBFIL (filename)	This is designated at the INSERT column list and loads LOBs from external file name source. The data itself contains the filename with a full path name, otherwise	1. LOBS FROM pathnames 2. MODIFIED BY LOBSINFILE	Pathnames where LOB files will be found. This must be coded to activate the LOBS FROM clause. The data itself contains the filename without a full path name as the

LOBs are searched in the same directory as the LOAD script.

LOBS FROM path will be searched.

DB2 LOAD command

Take a look at the syntax of the DB2 LOAD command:

Listing 2. The syntax of the DB2 LOAD command

```

.-,-----
V |
>>-LOAD-----+--FROM-----+filename-----+--OF--filetype----->
'-CLIENT-'          +-pipename---+
+-device-----+
'-cursorname-'

>+-----+-----+-----+-----+-----+-----+-----+----->
|               V |               V |
|'-LOBS FROM---lob-path---'| '-XML FROM---xml-path---'|
|
|               V |
|'-MODIFIED BY---file-type-mod---'|
|
|-----+-----+-----+-----+-----+-----+-----+-----+----->
|               V |
|'-METHOD---+L--(---column-start--column-end---)---+-----+-----+-----+-----+-----+-----+-----+-----+----->
|               |               V |
|               |'-NULL INDICATORS--(---null-indicator-list---)-'|
|               V |
|+N--(---column-name---)-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
|               V |
|'-P--(---column-position---)-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
|
|-----+-----+-----+-----+-----+-----+-----+-----+----->
|'-XMLPARSE---+STRIP---+WHITESPACE-'
|'-PRESERVE-'
|
|-----+-----+-----+-----+-----+-----+-----+-----+----->
|'-XMLVALIDATE USING---+XDS---+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
| |'-DEFAULT---schema-sqlid-' |
|+SCHEMA---schema-sqlid-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
|'-SCHEMALOCATION HINTS-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
|
|-----+-----+-----+-----+-----+-----+-----+-----+----->
|'-SAVECOUNT--n-' '-ROWCOUNT--n-' '-WARNINGCOUNT--n-'
|
|-----+-----+-----+-----+-----+-----+-----+-----+----->
|'-MESSAGES--message-file-'
|
|-----+-----+-----+-----+-----+-----+-----+-----+----->
|'-TEMPFILES PATH--temp-pathname-'

```

```
>--+--INSERT-----+----->
|      .-KEEPDICTIONARY--. |
+-REPLACE-----+-----+
|      '-RESETDICTIONARY-' |
+-RESTART-----+-----+
|'-TERMINATE-----'
|
>--INTO--table-name--+----->
|      v ,-----|
|'-(----insert-column-+--)-'
|
>--+-----+----->
|      v ,-----|
|      v (1) (2) |
+-FOR EXCEPTION--table-name-----+--+
+-NORANGEEXC--+
|'-NONUNIQUEEXC-'
|
>--+-----+----->
|'-STATISTICS--+USE PROFILE--+
|'-NO-----'
|
>--+-----+----->
|      .-NO-----|
+-COPY--+YES--+USE TSM--+-----+--+
|      |          '-OPEN--num-sess--SESSIONS-' |
|      v ,-----|
|      +TO---device/directory+-----+
|      '-LOAD--lib-name-----+
|      |          '-OPEN--num-sess--SESSIONS-' |
|'-NONRECOVERABLE-----'
|
>--+-----+-----+----->
|'-WITHOUT PROMPTING-' '-DATA BUFFER--buffer-size-'
|
>--+-----+-----+----->
|'-SORT BUFFER--buffer-size-' '-CPU_PARALLELISM--n-'
|
>--+-----+-----+----->
|'-DISK_PARALLELISM--n-' |          .-YES-. |
|'-FETCH_PARALLELISM--+NO--+
|
>--+-----+-----+----->
|'-INDEXING MODE--+AUTOSELECT--+
+-REBUILD-----+
+-INCREMENTAL+
|'-DEFERRED----'
|
.-ALLOW NO ACCESS-----
>--+-----+----->
|'-ALLOW READ ACCESS--+-----+
|'-USE--tablespace-name-'
|
>--+-----+----->
|'-SET INTEGRITY PENDING CASCADE--+IMMEDIATE--+
|'-DEFERRED--'
|
>--+-----+----->
|'-LOCK WITH FORCE-'
|
>--+-----+-----+----->
|'-SOURCEUSEREXIT--executable-- Redirect Input/Output parameters |--+-----+
|'-PARALLELIZE-'
|
>--+-----+-----><
|      .-PARTITIONED DB CONFIG-. v |
```

```
--+-----partitioned-db-option+-'

Ignore and Map parameters

|--+----->
|      v , ----- |
|'-IGNORE--(---schema-sqlid+---)-'|
>--+-----|
|      v , ----- |
|'-MAP--(---(--schema-sqlid--,--schema-sqlid--)++-)-'|

Redirect Input/Output parameters

|--+-----+----+
|'-REDIRECT--+INPUT FROM---BUFFER--input-buffer-+-+-----+----+'
|      '-FILE--input-file-----'   '-OUTPUT TO FILE--output-file-' |
|'-OUTPUT TO FILE--output-file-----'
```

Perl Script Converter from SQL*Loader to DB2 LOAD

The Perl script given here is used to convert a SQL*Loader control file into the equivalent DB2 LOAD script. The SQL*Loader can be invoked in conventional and direct path mode, but SQL strings are not used in direct path mode by the SQL*Loader to format the data. Oracle DBAs sometimes overlook those options and use conventional and direct path interchangeably without realizing the side effects. At a basic level, the SQL*Loader conventional path is equivalent to the DB2 IMPORT utility, which uses the engine to do the bulk inserts in DB2. By using the database engine, you can take advantages of the triggers and more, which is not the case either in SQL*Loader direct path or in the DB2 LOAD utility. The Perl script converts both direct and conventional path versions of the SQL*Loader control scripts to DB2 LOAD scripts. If you wish to use the DB2 IMPORT as an equivalent of the SQL*Loader conventional path, modify the keyword from LOAD to IMPORT in the generated file.

Prerequisite to run the Perl script

You need to have Perl installed on your target platform to convert `SQL*Loader` scripts to `DB2 LOAD` scripts. It is assumed that you are familiar with the installation of the Perl tool on your target platform. The examples shown here are for Windows platform but you can use the tool on any platform of your choice.

How to run Perl script

When you run the tool without specifying any argument, it shows the usage of the tool, as shown in Listing 3 below:

Listing 3. Usage of the tool


```
C:\>perl ora2db2.pl
USAGE: perl ora2db.pl -c controlfile [options]
      -o ostype (Unix|Windows - default is Unix. Other value is Windows
      -m message_directory_name (default is MSG_DIR)
      -e dump_directory_name (default is DUMP_DIR).
          This dir should reside on all partitions on DB2 server.
      -d data_file_name (default is INPUT_FILE)
      -f defaultif clause set to either (default|null - default is null)
      -s schema name. Replace with the schema name in control file
      -g timestamp format. Default is YYYY-MM-DD
```

You can specify some parameters as an argument to the SQLLDR command and they override options if defined in the control file. This may lead to potential migration problems if you are using a script to call SQLLDR with arguments. Those arguments can be matched with the switches given in the usage of the tool so that DB2 LOAD script is created with correct arguments.

The following examples show how to convert direct and conventional path control files with different format options.

```
C:\>perl ora2db2.pl -c test1.ctl -d data\test1.data -m msg -e dump -o Windows
                        -s ADMIN -f null > load1.db2
C:\>perl ora2db2.pl -c test2.ctl -d data\test2.data -m msg -e dump -o Windows
                        -s ADMIN -f null > load2.db2
```

How to run Perl and modify the script:

- It is not necessary to know Perl in order to convert your SQL*Loader scripts to DB2 LOAD scripts.
- But, at times, you might feel a need to modify the tool as per your requirement.
- There are possibilities when this tool may fail to parse the SQL*Loader script if it is too complex.
- If you are comfortable working with Perl scripts, you can follow these simple guidelines to easily modify and work with given Perl script.
 1. Install Perl tool on your Windows or your development machine.
 2. Install Eclipse and also install a plug-in for Perl.
 3. Create a Perl project and copy ora2db.pl and userexit.pl files to your project.
 4. You can use Perl in debugger mode to go through step by step to modify / enhance the code as per your need.
- We will appreciate if you let us know what changes did

you make so that it is useful to others also.

- The tool is provided as is without any warranty whatsoever. Please read the license agreement for details.

Listing 4. SQL*Loader test1.ctl using direct path and positional columns

```
UNRECOVERABLE
LOAD DATA
INFILE 'INPUT_FILE'
APPEND
INTO TABLE JOHN.TABLE1
TRAILING NULLCOLS
(
  BC_OFF_BCBANK POSITION(2:4) CHAR      ,
  BC_OFF_SEGMENT_TYPE POSITION(5:12) CHAR "rtrim(:BC_OFF_SEGMENT_TYPE,' ')"      ,
  BC_OFF_NUM POSITION(13:18) CHAR DEFAULTIF BC_OFF_NUM= ' ??????',
  BC_OFF_SQ_EFF_DATE POSITION(19:26) CHAR DEFAULTIF BC_OFF_SQ_EFF_DATE=" ????????",
  BC_OFF_SQ_ENT_DATE POSITION(27:40) CHAR DEFAULTIF BC_OFF_SQ_ENT_DATE = ' ??????????????',
  BC_OFF_DELETE POSITION(41:41) CHAR,
  BC_OFF_EFF_DATE POSITION(42:53) CHAR DEFAULTIF BC_OFF_EFF_DATE = ' ??????????????',
  BC_OFF_INITIALS POSITION(54:56) CHAR "rtrim(:BC_OFF_INITIALS,' ')",
  BC_OFF_NAME POSITION(57:76) CHAR "rtrim(:BC_OFF_NAME,' ')",
  BC_OFF_C_L_SECTION POSITION(77:78) CHAR,
  BC_OFF_PHONE_NR POSITION(79:90) CHAR DEFAULTIF BC_OFF_PHONE_NR=' ??????????????',
  BC_OFF_SC_LND_LMT POSITION(91:98) CHAR DEFAULTIF BC_OFF_SC_LND_LMT = ' ??????????',
  BC_OFF_UNSC_LND_LMT POSITION(99:106) CHAR DEFAULTIF BC_OFF_UNSC_LND_LMT = ' ?????????',
  BC_OFF_NEWCOL POSITION(107:107) CHAR NULLIF (BC_OFF_NEWCOL=BLANKS),
  PRCS_DTE CONSTANT "PROCESSDATE",
  PRCS_YR_MTH_NBR CONSTANT "PROCYRMTH"
)
```

Listing 5. Converted DB2 LOAD scripts using method L

```
-- Converting Oracle SQL*Loader Control File test1.ctl to DB2

-- ALTER Statements to take care of CONSTANT parameters
ALTER TABLE TABLE2 ALTER COLUMN PRCS_DTE SET WITH DEFAULT 'PROCESSDATE';
ALTER TABLE TABLE2 ALTER COLUMN PRCS_YR_MTH_NBR SET WITH DEFAULT 'PROCYRMTH';

-- DB2 LOAD Script
LOAD FROM "data\test1.data"
OF ASC
MODIFIED BY ANYORDER USEDEFAULTS STRIPTBLANKS TIMESTAMPFORMAT="YYYY-MM-DD"
DUMPFILE="dump\admin_table1.dump"
METHOD L
(
  2 4
  ,5 12
  ,13 18
  ,19 26
  ,27 40
  ,41 41
  ,42 53
  ,54 56
  ,57 76
  ,77 78
  ,79 90
```

```

,91 98
,99 106
,107 107
)
MESSAGES "msg\admin_table1.msg"
INSERT INTO "ADMIN"."TABLE1"
( BC_OFF_BCBANK
,BC_OFF_SEGMENT_TYPE
,BC_OFF_NUM
,BC_OFF_SQ_EFF_DATE
,BC_OFF_SQ_ENT_DATE
,BC_OFF_DELETE
,BC_OFF_EFF_DATE
,BC_OFF_INITIALS
,BC_OFF_NAME
,BC_OFF_C_L_SECTION
,BC_OFF_PHONE_NR
,BC_OFF_SC_LND_LMT
,BC_OFF_UNSC_LND_LMT
,BC_OFF_NEWCOL
)
NONRECOVERABLE
INDEXING MODE AUTOSELECT
;

-- UPDATE Statements for setting proper DEFAULTIF parameters
UPDATE "ADMIN"."TABLE1"
SET BC_OFF_NUM = NULL
WHERE BC_OFF_NUM = ' ??????';

UPDATE "ADMIN"."TABLE1"
SET BC_OFF_SQ_EFF_DATE = NULL
WHERE BC_OFF_SQ_EFF_DATE = ' ????????';

UPDATE "ADMIN"."TABLE1"
SET BC_OFF_SQ_ENT_DATE = NULL
WHERE BC_OFF_SQ_ENT_DATE = ' ??????????????';

UPDATE "ADMIN"."TABLE1"
SET BC_OFF_EFF_DATE = NULL
WHERE BC_OFF_EFF_DATE = ' ??????????????';

UPDATE "ADMIN"."TABLE1"
SET BC_OFF_PHONE_NR = NULL
WHERE BC_OFF_PHONE_NR = ' ??????????????';

UPDATE "ADMIN"."TABLE1"
SET BC_OFF_SC_LND_LMT = NULL
WHERE BC_OFF_SC_LND_LMT = ' ????????';

UPDATE "ADMIN"."TABLE1"
SET BC_OFF_UNSC_LND_LMT = NULL
WHERE BC_OFF_UNSC_LND_LMT = ' ????????';

-- UPDATE Statements for setting proper NULLIF parameters
UPDATE "ADMIN"."TABLE1"
SET BC_OFF_NEWCOL = NULL
WHERE BC_OFF_NEWCOL = 'BLANKS)';

```

Listing 6. SQL*Loader test2.ctl using conventional path and delimited columns

```

LOAD DATA
INFILE 'INPUT_FILE'

```

```

APPEND
INTO TABLE JOHN.TABLE2
FIELDS TERMINATED BY ' '
TRAILING NULLCOLS
(
CR_BUR_PTFLO_TYP_DESC char(255) NULLIF CR_BUR_PTFLO_TYP_DESC=BLANKS,
DW_PROD_SERV_FEE_PLN_RCD_ID,
DW_ULT_PROD_SERV_ID,
ACCT_NBR,
ACCT_GRP_NBR,
APPL_CDE,
PRCS_GRP_NBR,
FEE_CAT_CDE,
FEE_PLN_NBR,
FEE_DESC,
FEE_TYP_CDE,
FEE_EARN_CDE,
CMPT_1_NXT_ASSMT_DTE      DATE "YYYY-MM-DD",
CMPT_2_NXT_ASSMT_DTE      DATE "YYYY-MM-DD",
CMPT_3_NXT_ASSMT_DTE      DATE "YYYY-MM-DD",
CUR_PMT_DUE_DTE           DATE "YYYY-MM-DD",
EARN_GOOD_THRU_1_DTE      DATE "YYYY-MM-DD",
EARN_GOOD_THRU_2_DTE      DATE "YYYY-MM-DD",
PR_PMT_DUE_DTE            DATE "YYYY-MM-DD",
NXT_PMT_DUE_DTE           DATE "YYYY-MM-DD",
DW_ASP_ID,
CLNT_ID,
CUR_REC_IND,
SOR_EXP_DTE               "NVL(:SOR_EXP_DTE,'4444-12-31')",
EFF_DTE                   DATE "YYYY-MM-DD"
)

```

Listing 7. Converted DB2 LOAD scripts using method P

```

-- Converting Oracle SQL*Loader Control File test2.ctl to DB2

-- DB2 LOAD Script
LOAD FROM "data\test2.data"
OF DEL
MODIFIED BY COLDEL0x09 ANYORDER USEDEFAULTS TIMESTAMPFORMAT="YYYY-MM-DD"
DUMPFILE="dump\admin_table2.dump"
METHOD P
(
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
)
MESSAGES "msg\admin_table2.msg"
INSERT INTO "ADMIN"."TABLE2"
( CR_BUR_PTFLO_TYP_DESC
,DW_PROD_SERV_FEE_PLN_RCD_ID
,DW_ULT_PROD_SERV_ID
,ACCT_NBR
,ACCT_GRP_NBR
,APPL_CDE
,PRCS_GRP_NBR
,FEE_CAT_CDE
,FEE_PLN_NBR
,FEE_DESC
,FEE_TYP_CDE
,FEE_EARN_CDE
,CMPT_1_NXT_ASSMT_DTE
,CMPT_2_NXT_ASSMT_DTE
,CMPT_3_NXT_ASSMT_DTE
,CUR_PMT_DUE_DTE
,EARN_GOOD_THRU_1_DTE
,EARN_GOOD_THRU_2_DTE

```

```

,PR_PMT_DUE_DTE
,NXT_PMT_DUE_DTE
,DW_ASP_ID
,CLNT_ID
,CUR_REC_IND
,SOR_EXP_DTE
,EFF_DTE
)
NONRECOVERABLE
INDEXING MODE AUTOSELECT
;

-- UPDATE Statements for setting proper NULLIF parameters
UPDATE "ADMIN"."TABLE2"
SET CR_BUR_PTFLO_TYP_DESC = NULL
WHERE TRIM(CR_BUR_PTFLO_TYP_DESC) = '';

UPDATE "ADMIN"."TABLE2"
SET SOR_EXP_DTE = COALESCE(SOR_EXP_DTE,'4444-12-31-00.00.00');

```

Notice that the `CONSTANT` columns were converted to `ALTER TABLE` statement with `DEFAULT` clause since DB2 does not allow `CONSTANT` keyword in DB2 LOAD script. DB2 LOAD will apply `DEFAULT` value while loading the data whereas SQL*Loader direct path does not apply default value.

Also notice the use of SQL strings in `UNRECOVERABLE` option (using direct path) in Oracle control file test1.ctl example but those SQL strings will be ignored for the direct path. The converted equivalent update statements can be ignored in DB2.

Did you notice that above conversion produced some `UPDATE` statements that will massage the data as per SQL Strings applied? If you do not like to apply these `UPDATE` statements due to performance reasons on a very large table, you may choose another option of massaging the data file using `SOURCEUSEREXIT` and the following section explains this.

USEREXIT for DB2 LOAD

One of the major architectural differences between DB2 and Oracle is how empty strings are handled. Oracle treats empty strings as `NULL`, whereas DB2 does not. This holds true when loading data using `LOAD` utility and positional columns are used in SQL*Loader control file. If those columns are empty, DB2 treats them as empty, whereas Oracle takes them as `NULL`.

The following sample data has `NULL` indicators at position 23 and 32 for data defined at position 24:27 and 28:31:

Listing 8. NULL indicators

```
FILE1 has 7 elements:
```

```

ELE1 positions 01 to 20
ELE2 positions 21 to 22
ELE3 positions 23 to 23
ELE4 positions 24 to 27
ELE5 positions 28 to 31
ELE6 positions 32 to 32
ELE7 positions 33 to 40

1...5...10...15...20...25...30...35...40
Test data 1      XXN 123abcdN
Test data 2 and 3  QQY      wxyzN
Test data 4,5 and 6 WWN6789    Y

```

The following LOAD script using NULL INDICATORS option will treat a column as NULL if null indicator is set to Y:

Listing 9. NULL INDICATORS option

```

TABLE has 5 columns:

COL1 VARCHAR 20 NOT NULL WITH DEFAULT
COL2 SMALLINT
COL3 CHAR 4
COL4 CHAR 2 NOT NULL WITH DEFAULT
COL5 CHAR 2 NOT NULL

db2 load from file1 of asc modified by striptblanks reclen=40
method L (1 20, 21 22, 24 27, 28 31)
null indicators (0,0,23,32)
insert into table1 (col1, col5, col2, col3)

```

Take example of Oracle control script **test1.ctl**, which was converted using our Perl script. DB2 LOAD requires specifying NULL INDICATORS to specifically load null values. This will require adding NULL indicators for all columns in the data file for each record. For such conditions, DB2 LOAD SOURCEUSEREXIT option can be used to massage the data file as per logic in the user exit program. This approach does not require to process the file before running the LOAD. Using SOURCEUSEREXIT option allows DB2 LOAD to read the data file and transfer the data record to the user exit program which can process it and hand it over back again to the DB2 LOAD.

Listing 10. Modified DB2 LOAD script to use SOURCEUSEREXIT

```

LOAD FROM data\test1.data
OF ASC
MODIFIED BY ANYORDER USEDEFAULTS STRIPTBLANKS TIMESTAMPFORMAT="YYYY-MM-DD"
DUMPFIL="dump\admin_table1.dump"
METHOD L
(
  2 4
  ,5 12
  ,13 18
  ,19 26
  ,27 40
  ,41 41

```

```

,42 53
,54 56
,57 76
,77 78
,79 90
,91 98
,99 106
,107 107
)
NULL INDICATORS (108,109,110,111,112,113,0,114,115,116,117,118,119,120)
MESSAGES "msg\admin_table1.msg"
INSERT INTO "ADMIN"."TABLE1"
( BC_OFF_BCBANK
,BC_OFF_SEGMENT_TYPE
,BC_OFF_NUM
,BC_OFF_SQ_EFF_DATE
,BC_OFF_SQ_ENT_DATE
,BC_OFF_DELETE
,BC_OFF_EFF_DATE
,BC_OFF_INITIALS
,BC_OFF_NAME
,BC_OFF_C_L_SECTION
,BC_OFF_PHONE_NR
,BC_OFF_SC_LND_LMT
,BC_OFF_UNSC_LND_LMT
,BC_OFF_NEWCOL
)
NONRECOVERABLE
INDEXING MODE AUTOSELECT
SOURCEUSEREXIT userexit.pl redirect input from buffer 'colninds:(2 4,5 12,
13 18,19 26,27 40,41 41,42 53,54 56,57 76,77 78,79 90,91 98,99 106,
107 107) nullinds:(108,109,110,111,112,113,0, 114,115,116,117,118,
119,120)'
                                output to file userexit.log
;

```

Note:

1. This DB2 LOAD script (as shown left hand side) was not generated by the Perl script and hand changes were made as shown in **BOLD**. This example shows how SOURCEUSEREXIT option can be used in DB2 LOAD script to modify the data file at run time. This becomes very helpful when the data file size is in several GB or TB and having twice the space of the file is not an option.
2. The SOURCEUSEREXIT argument in the LOAD command takes a Perl script **userexit.pl** as a source user exit program and this program massages the data file using the buffer string (passed to it through LOAD) to add NULL indicators at the end of the each record so that DB2 can handle NULLs appropriately.
3. The Perl script (userexit.pl) to modify the data at run time should be kept in **sqllib** directory on the server so that LOAD can find it.
4. You can have log statements in **userexit.pl** script and they can be put to a file using **OUTPUT TO FILE** option.
5. The Perl script should process the arguments in a certain

fashion and refer to the DB2 documentation or `userexit.pl` for details.

Conclusion

It is relatively easy for Oracle DBAs to take their knowledge of the `SQL*Loader` and apply it to learning the DB2 `LOAD` and `IMPORT` utilities. The focus of this article was specifically on `LOAD`, but much of this information applies to `IMPORT` as well. This article has shown you the main keywords to both `SQL*Loader` and DB2 `LOAD`, and how they compare. This article has also provided an easy-to-use Perl script that can convert most of the `SQL*Loader` scripts you may have. This should make things much easier for you in your adoption of using DB2 in your organization.

Acknowledgement

The authors of this article would like to thank David Sciaraffa of DB2 Toronto lab for providing the `userexit.pl` program as an example to process the records to add `NULL` indicators.

Downloads

Description	Name	Size	Download method
Perl scripts to migrate SQL*Loader to DB2 Load	DB2LOAD.zip	10KB	HTTP

[Information about download methods](#)

Resources

Learn

- [IBM DB2 Database for Linux, UNIX, and Windows Information Center](#): Learn about DB2 9.5.
- [db2ude](#): Check out Vikram's blog, where he shares information, tips, and techniques on DB2.
- [developerWorks Information Management zone](#): Learn more about Information Management. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with [developerWorks technical events and webcasts](#).
- [Technology bookstore](#): Browse for books on these and other technical topics.

Get products and technologies

- [DB2 9.5 Enterprise Edition](#): Download a free trial version of DB2 9.5 Enterprise Edition.
- [DB2 9.5 Express-C](#): Download free license of DB2 9.5 Express-C.
- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.

About the authors

Burt L. Vialpando

Burt Vialpando is a nine-year IBM employee and a certified IT specialist currently working for the SMPO team performing presales Oracle to DB2 migration support. With over 25 years of IT experience, he holds numerous DB2, Oracle, and other certifications. He currently serves on the Certification Board, the Competency Team and the Migrations Committee.

Vikram S. Khatri

Vikram Khatri works for IBM in the Sales and Distribution Division and is a member of DB2 Migration team. Vikram has 21 years of IT experience and specializes in migrating non-DB2 databases to DB2. Vikram supports the DB2 technical sales organization by assisting with complex database migration projects as well as with database performance benchmark testing.

Trademarks

Oracle 11g is a trademark of Oracle Corporation.
DB2 9.5 is a trademark of IBM Corporation.