

# Oracle (PL/SQL)

Lesson 11: SQL \* Loader

## Lesson Objectives

- To understand the following topics:
  - Outline of SQL\*Loader, an Oracle-supplied utility
  - SQL \* Loader Environment



11.1: SQL\*Loader?

## What Is SQL \* Loader?

- SQL\*Loader is a bulk loader utility used for moving data from external files into the Oracle database.
- SQL\*Loader supports various load formats, selective loading, and multi-table loads.



Copyright © Capgemini 2015. All Rights Reserved 3

### Note:

SQL\*Loader has a powerful data parsing engine that puts little limitation on the format of the data in the datafile. Thus SQL\*Loader supports various load formats, selective loading, and multi-table loads.

11.2: SQL\*Loader as a Utility

## Usage Of SQL \* Loader

- SQL\*Loader can be used for the following utilities:
  - To load data from multiple datafiles during the same load session.
  - To load data into multiple tables during the same load session.
  - To specify the character set of the data.
  - To selectively load data (you can load records based on the records' values).
  - To manipulate the data before loading it, using SQL functions.
  - To generate unique sequential key values in specified columns.

11.2: SQL\*Loader as a Utility

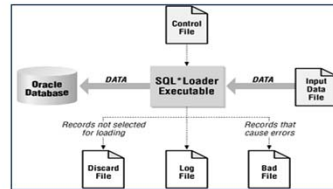
## Usage Of SQL \* Loader

- To use the operating system's file system to access the datafiles
- To load data from disk, tape, or named pipe
- To generate sophisticated error reports, which greatly aids troubleshooting
- To load arbitrarily complex object-relational data
- To use secondary datafiles for loading LOBs and collections

## 11.3: SQL\* Loader Environment

## SQL \* Loader Environment


- A typical SQL\*Loader session takes a control file as input. The input file controls the behavior of SQL\*Loader and one or more datafiles.
- The output of SQL\*Loader is an Oracle database (where the data is loaded), a log file, a bad file, and potentially a discard file.



11.3:SQL\* Loader Environment

## The Control File

- The SQL\*Loader Control file is the text file which is a key to any load process.
- The Control file provides the following information to Oracle for data load:
  - Datafile name, location, and format
  - Character sets used in the datafiles
  - Datatypes of fields in those files
  - Information on how each field is delimited
  - Information on the tables and columns to be loaded

 Copyright © Capgemini 2015. All Rights Reserved 7

### The Control File:

Some of the items shown in the list on the slide may also be passed to SQL\*Loader as command-line parameters.

For example: The name and location of the input file, may be passed on the command line instead of in the Control file. The same goes for the names and locations of the Bad files and the Discard files.


It is also possible for the Control file to contain the actual data to be loaded. This is sometimes done when small amounts of data needs to be distributed to many sites, because it reduces the number of files that need to be passed around (to just one file).

If the data to be loaded is contained in the Control file, then there is no need for a separate Data file.

11.3:SQL\* Loader Environment

## The Control File

- The SQL\*Loader Control file is the text file which is a key to any load process.
- The Control file provides the following information to Oracle for data load:
  - Datafile name, location, and format
  - Character sets used in the datafiles
  - Datatypes of fields in those files
  - Information on how each field is delimited
  - Information on the tables and columns to be loaded

 Copyright © Capgemini 2015. All Rights Reserved 8

### The Control File:

Some of the items shown in the list on the slide may also be passed to SQL\*Loader as command-line parameters.

For example: The name and location of the input file, may be passed on the command line instead of in the Control file. The same goes for the names and locations of the Bad files and the Discard files.

It is also possible for the Control file to contain the actual data to be loaded. This is sometimes done when small amounts of data needs to be distributed to many sites, because it reduces the number of files that need to be passed around (to just one file).

If the data to be loaded is contained in the Control file, then there is no need for a separate Data file.



11.3: SQL\* Loader Environment

## The Log File

- The log file is a record of SQL\*Loader's activities during a load session.
- The log file contains information as given below:
  - The names of the control file, log file, bad file, discard file, and data file
  - The values of several command-line parameters
  - A detailed breakdown of the fields and datatypes in the data file that was loaded
  - Error messages for records that cause errors
  - Messages indicating when records have been discarded



Copyright © Capgemini 2015. All Rights Reserved 9

**Note:**

Always review the Log file after a load to ensure that no errors have occurred, or at least that no unexpected errors occurred.

This type of information is written to the Log file. However, the information is not displayed on the terminal screen.

## The Log File

- A summary of the load that includes the following:
- The number of logical records read from the data file
- The number of rows rejected because of errors
- The number of rows discarded because of selection criteria
- The time elapsed for the load

11.3: SQL\* Loader Environment

## The Bad File & Discard File

- Whenever you insert data into a database, you run the risk of that insert failing because of some type of error.
- Integrity constraint violations undoubtedly represent the most common type of error.
  - However, other problems, such as the lack of free space in a tablespace, can also cause insert operations to fail.
- Whenever SQL\*Loader encounters a database error while trying to load a record, it writes that record to a file known as the “Bad file”.

## The Bad File & Discard File

- Discard files are used to hold records that do not meet selection criteria specified in the SQL\*Loader control file.
  - By default, SQL\*Loader will attempt to load all the records contained in the input file.
- Records that do not meet the specified criteria are not loaded, and are instead written to a file known as the "Discard file".



Copyright © Capgemini 2015. All Rights Reserved 12

### Note:

You have the option in your Control file, of specifying selection criteria that a record must meet before it is loaded.

### Comparison between Bad Files and Discard Files:

Discard files are optional. You will only get a Discard file if you have specified a Discard file name, and if at least one record is actually discarded during the load.

Bad files are not optional. The only way to avoid having a Bad file generated is to run a load that results in no errors. If even one error occurs, then SQL\*Loader will create a Bad file and write the offending input record (or records) to that file.

The format of your Bad files and Discard files will exactly match the format of your input files. This scenario occurs because SQL\*Loader writes the exact records, which cause errors, or those that are discarded, to those files.

If you are running a load with multiple input files, you will get a distinct set of bad files and discard files for each input file.

11.4: Invoking SQL\*Loader

## Methods Of Invoking SQL\*Loader

- SQL\*Loader can be invoked in one of the following three ways:
  - sqlldr
  - sqlldr keyword=value [keyword=value ...]
  - sqlldr value [value ...]

**Invoking SQL\*LOADER:**

On Unix systems, the command used to invoke SQL\*Loader is `sqlldr`. On Windows systems running Oracle8i, release 8.1 or higher, the command is also `sqlldr`.

Prior to release 8.1, the SQL\*Loader command on Windows systems included the first two digits of the Oracle release number. Thus you had `sqlldr80` (Oracle8, release 8.0), `sqlldr73` (Oracle7, release 7.3), and so forth.

SQL\*Loader can be invoked in one of three ways:

`sqlldr`

`sqlldr keyword=value [keyword=value ...]`

`sqlldr value [value ...]`

Issuing the `sqlldr` command by itself results in a list of valid command-line parameters being displayed. Command-line parameters are usually keyword / value pairs, and may be any combination of the following:

`USERID={username[/password][@net_service_name]}|}`

`CONTROL=control_file_name`

`LOG=path_file_name`

`BAD=path_file_name`

`DATA=path_file_name`

`DISCARD=path_file_name`

`DISCARDMAX=logical_record_count`

`SKIP=logical_record_count`

`SKIP_INDEX_MAINTENANCE={TRUE | FALSE}`

`SKIP_UNUSABLE_INDEXES={TRUE | FALSE}`

`LOAD=logical_record_count`

`ERRORS=insert_error_count`

`ROWS=rows_in_bind_array`

`BINDSIZE=bytes_in_bind_array`

`SILENT=[(keyword[,keyword...])]`

`DIRECT={TRUE | FALSE}`

`PARFILE=path_file_name`

`PARALLEL={TRUE | FALSE}`

`READSIZE=bytes_in_read_buffer`

`FILE=database_datafile_name`

Command-line parameters may be passed by position instead of by keyword. Command-line parameters are discussed in detail in the subsequent slides.

## Invoking SQL \* Loader

■ Given below is a list of Command Line Parameters:

- **USERID = {username[/password] [@net\_service\_name]]/}**
  - Specifies the username and password to use when connecting to the database. The net\_service\_name parameter optionally allows you to connect to a remote database. Use a forward-slash character ( / ) to connect to a local database by using operating system authentication.
- **CONTROL = control\_file\_name**
  - Specifies the name, which may include the path, of the control file. The default extension is .ctl.
- **LOG = path\_file\_name**
  - Specifies the name of the Log file to generate for a load session. You may include a path as well. By default, the Log file takes on the name of the Control file, but with a .log extension. The Log file is written to the same directory as the Control file. If you specify a different name, the default extension is still .log.



Copyright © Capgemini 2015. All Rights Reserved 15

### Invoking SQL\*Loader:

**USERID = {username[/password] [@net\_service\_name]]/}**

#### Note:

On Unix systems, you may want to omit the password and allow SQL\*Loader to prompt you for it. If you omit both the username and the password, SQL\*Loader will prompt you for both.

On Unix systems you should generally avoid placing a password on the command line, because that password will be displayed whenever other users issue a command, such as ps -ef, that displays a list of current processes running on the system. Either let SQL\*Loader prompt you for your password, or use operating system authentication. (If you don't know what operating system authentication is, ask your DBA.)

## Invoking SQL \* Loader

- **BAD = path\_ file\_name**
  - Specifies the name of the Bad file. You may include a path as part of the name. By default, the Bad file takes the name of the Control file, but with a .bad extension. The Bad file is written to the same directory as the Control file.
- **DATA = path\_ file\_name**
  - Specifies the name of the file containing the data to load. You may include a path as part of the name. By default, the name of the Control file is used, but with the .dat extension. If you specify a different name, the default extension is still .dat



Copyright © Capgemini 2015. All Rights Reserved 15

Invoking SQL\*Loader (contd.):

**LOG = path\_ file\_name**

Note: If you use the LOG parameter to specify a name for the log file, it will no longer be written automatically to the directory that contains the control file.

**BAD = path\_ file\_name**

Note: If you specify a different name, the default extension remains .bad. However, if you use the BAD parameter to specify a bad file name, the default directory becomes your current working directory. If you are loading data from multiple files, then this bad file name only gets associated with the first file being loaded.



## Invoking SQL \* Loader

- DISCARD = path\_file\_name
  - Specifies the name of the Discard file. You may include a path as part of the name. By default, the Discard file takes the name of the Control file, but it has a .dis extension. If you specify a different name, the default extension is still .dis.
- DISCARDMAX = logical\_record\_count
  - Sets an upper limit on the number of logical records that can be discarded before a load will terminate. The limit is actually one less than the value specified for DISCARDMAX.



Copyright © Capgemini 2015. All Rights Reserved 17

Invoking SQL\*Loader (contd.):

DATA = path\_file\_name

Note: If you are loading from multiple files, you can only specify the first file name using this parameter. Place the names of the other files in their respective INFILE clauses in the control file.

DISCARD = path\_file\_name

Note: If you are loading data from multiple files, then this discard file name only gets associated with the first file being loaded.

## Invoking SQL \* Loader

- SKIP = logical\_record\_count
  - Allows you to continue an interrupted load by skipping the specified number of logical records.
- LOAD = logical\_record\_count
  - Specifies a limit on the number of logical records to load. The default is to load all records. Since LOAD only accepts numeric values, it is not possible to explicitly specify the default behavior.
- ROWS = rows\_in\_bind\_array
  - The precise meaning of this parameter depends on whether you are doing a direct path load or a conventional load.



Copyright © Capgemini 2015. All Rights Reserved 18

Invoking SQL\*Loader (contd.):

DATA = path\_ file\_name

Note: If you are loading from multiple files, you can only specify the first file name using this parameter. Place the names of the other files in their respective INFILE clauses in the control file.

DISCARD = path\_ file\_name

Note: If you are loading data from multiple files, then this discard file name only gets associated with the first file being loaded.

Invoking SQL\*Loader (contd.):

You can even mix the positional and keyword methods of passing command-line parameters. The rule to be followed while doing this is that all positional parameters must come first. Once you start using keywords, you must continue to do so.

For example:

```
sqlldr system/manager control=profile.ctl log=profile.ctl
```

When you pass parameters positionally, you must not skip any. Also, be sure to get the order right.

You must supply parameter values in the order shown earlier in this section.

Given the fact that you typically will use only a few parameters out of the many that are possible, it's usually easier to pass those parameters as keyword/value pairs than it is to pass them positionally.

Using keyword/value pairs also makes long SQL\*Loader commands somewhat self-documenting. An exception to this rule is that you might wish to pass the username and password positionally, since they come first, and then pass in the rest of the parameters by name.

## SQL \* Loader - Examples

### ■ Example 1:

- `sqlldr username@server/password control=loader.ctl`
  - The sample control file (loader.ctl) will load an external data file containing delimited data:

```
load data
infile 'c:\data\mydata.csv'
into table emp fields terminated by ","
optionally enclosed by '"' ( empno, empname, sal, deptno )
```

### ■ The mydata.csv file may look like the sample shown below:

- 10001,"Scott Tiger", 1000, 40
- 10002,"Frank Naude", 500, 20

## SQL \* Loader - Examples

### ■ Example 2:

- Another sample Control file with in-line data formatted as fix length records is discussed in this example.
- The trick is to specify "\*" as the name of the data file, and use BEGINDATA to start the data section in the control file:

```
load data infile * replace
into table departments ( dept position (02:05) char(4), deptname position
(08:27) char(20) )
begindata
COSC COMPUTER SCIENCE
ENGL ENGLISH LITERATURE
MATH MATHEMATICS
POLY POLITICAL SCIENCE
```

# SQL \* Loader - Examples

- Example 1:
  - Step 1: Create the Relation
    - Create table customer in Oracle database Oracle account with the given structure.

Sql>Desc example

Name	Null ?	Type
-----		
custid		number
custname		varchar2(12)
age		number

## SQL \* Loader Case Examples

### ■ Step 2:

- Create Data File named customer.dat as shown below:

- 10,"AAA",20
- 20,"BBB",23
- 30,"CCC",41
- 40,"DDD",25

## SQL \* Loader Case Examples

- Step 3:

- Create sql loader control file named custcontrol.ctl with the given contents:

```
load data
infile 'd:\customer.dat'
into table customer
fields terminated by ',' optionally enclosed by '"'
(custid, custname, age)
```



## SQL \* Loader Case Examples

- Step 4:
  - At command prompt, run SQL Loader by entering the following details:

```
C:\sqlldr scott/tiger@oracle9i control=d:\custcontrol.ctl
```

```
userid/password@Host String
```

## Summary

- In this lesson, you have learnt:
  - SQL\*Loader as an Oracle utility
  - SQL\*Loader environment
    - Control File, Log File, Bad File, and Discard File
    - Invoking SQL\*Loader Environment
    - Command Line Parameters



## Review Question

- Question 1: We can use SQL\*Loader to load data into multiple tables during the same load session
  - True / False
- Question 2: A typical SQL\*Loader session takes as input a discard file
  - True/False
- Question 3: The SQL\*Loader control file is the text file which is a key to any load process.
  - True/False



## Review Question

- Question 4: The \_\_\_\_ file is a record of SQL\*Loader's activities during a load session.
- Question 5: SQL\*Loader Parameter \_\_\_\_ specifies a limit on the number of logical records to load.

