Oracle SQL Loader

Nota Preeliminar
Esta es una breve guía que pretende introducir nociones básicas en la utilización del utilitario SQL Loader. Para mayor información remitirse a las referencias consultadas para la elaboración de este documento que fueron principalmente la documentación en línea de Oracle y el libro Oreilly - Oracle Sql Loader The Definitive Guide.

Contenidos

Introducción:

SQL Loader es un utilitario que permite la inserción de datos desde un archivo plano a una o más bases de datos. Durante una sola de sus ejecuciones es posible llenar múltiples tablas con datos de múltiples archivos, manejar registros de ancho variable o fijo, manipular los datos entrantes para tratar con valores nulos, delimitadores y espacios en blanco, obviar registros o encabezados y reaccionar frente a fallas del proceso de cargado.

En la ilustración 1 podemos observar el funcionamiento de SQL Loader. Mediante el procesamiento de un archivo de control (Control File) que contiene esencialmente la localización de los archivos fuente, el formato de éstos y las tablas a ser llenadas, el ejecutable de la herramienta lee los datos de entrada, llena la base de datos y genera 3 tipos de archivos distintos: Discard File contiene los registros que no fueron cargados (por ejemplo debido a que en el archivo de control se haya configurado no registrar a las mujeres mayores de 25 años), Bad File contiene los registros que generaron errores (podría ser debido a fallas en el formato) y Log File, el archivo de log de la operación.

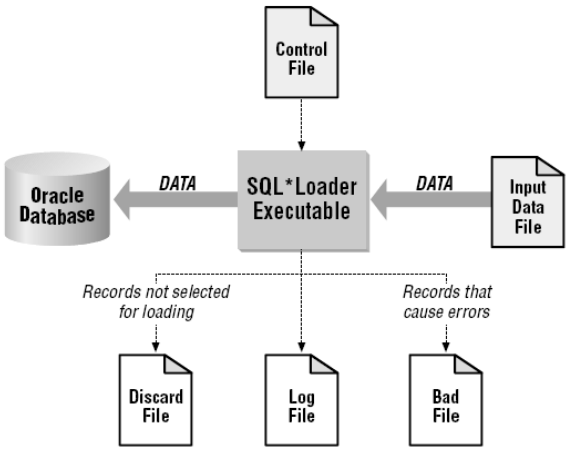

Ilustración 1 SQL Loader [1]

Un Ejemplo Rápido

En esta sección realizaremos un pequeño ejemplo que nos permitirá conocer un poco más sobre el funcionamiento de la herramienta y el modo en que se invoca. El objetivo es llenar la tabla Persona a partir de un archivo plano.

Preeliminares: Creación de la Tabla Persona

Antes de utilizar SQL Loader es necesario crear la tabla Persona en la base de datos, para ello debemos acceder al servidor Linux mediante SSH y una vez hecho esto, entrar a SQL Plus, un utilitario (al igual que SQL Loader) que permite la ejecución de comandos SQL, para correr el script de creación de la tabla.

1) Ingreso a SSH. En inicio/programas/conectividad/Secure Shell Client

2) Ingresar el nombre de usuario, nombre del servidor y cuando se solicite, la contraseña

3) Ingresar a SQL Plus escribiendo:

```
sqlplus NOMBREUSUARIO/CONTRASEÑA@NOMBRESERVICIO
```

Donde *NOMBRESERVICIO* es en nuestro caso *chie10*

4) Correr la sentencia de creación de la tabla.
Se debe copiar la siguiente sentencia a la consola de SQL PLus y oprimir ENTER para correrla. Al finalizar el proceso debe verse una pantalla similar a la ilustración 2.

```
CREATE TABLE  persona
(id_persona  NUMBER CONSTRAINT persona_pk PRIMARY KEY,
nombres  VARCHAR2(15) CONSTRAINT persona_nombres_nn NOT NULL,
apellidos VARCHAR2(15) CONSTRAINT persona_apellidos_nn NOT NULL,
nacionalidad VARCHAR2(30) CONSTRAINT persona_nacio_nn NOT NULL,
tipo_documento  VARCHAR(12) CONSTRAINT persona_tipo_dcto_nn NOT NULL,
numero_documento  VARCHAR(12) CONSTRAINT persona_num_dcto_nn NOT NULL,
CONSTRAINT persona_tipo_num_dcto_nd UNIQUE (tipo_documento, numero_documento),
fecha_nacimiento DATE CONSTRAINT persona_fecha_nac_nn NOT NULL,
sexo CHAR CONSTRAINT persona_sexo_nn NOT NULL
CONSTRAINT persona_sexo_cc CHECK (sexo IN ('f', 'm') )
) ;

CREATE SEQUENCE cons_persona;
```

```
SQL> CREATE TABLE  persona
  2  (id_persona  NUMBER CONSTRAINT persona_pk PRIMARY KEY,
  3  nombres  VARCHAR2(15) CONSTRAINT persona_nombres_nn NOT NULL,
  4  apellidos VARCHAR2(15) CONSTRAINT persona_apellidos_nn NOT NULL,
  5  nacionalidad VARCHAR2(30) CONSTRAINT persona_nacio_nn NOT NULL,
  6  tipo_documento  VARCHAR(12) CONSTRAINT persona_tipo_dcto_nn NOT NULL,
  7  numero_documento  VARCHAR(12) CONSTRAINT persona_num_dcto_nn NOT NULL,
  8  CONSTRAINT persona_tipo_num_dcto_nd UNIQUE (tipo_documento, numero_document
o),
  9  fecha_nacimiento DATE CONSTRAINT persona_fecha_nac_nn NOT NULL,
 10  sexo CHAR CONSTRAINT persona_sexo_nn NOT NULL
 11  CONSTRAINT persona_sexo_cc CHECK (sexo IN ('f', 'm') )
 12  ) ;

Tabla creada.
```

Ilustración 2 Aspecto al crear una tabla

Preeliminares: Creación del Archivo Plano

El siguiente paso es crear el archivo plano, la forma más directa de realizarlo es copiar los datos de la tabla 1 en Excel y después guardar el archivo en formato *.csv* (delimitado por comas).

Nombre	Apellido	Nacionalidad	Tipo Documento	Número Documento	Fecha Nacimiento	Sexo
Efraín	Medina	Colombiano	Cédula	1	25/05/1979	Masculino
León	De Grieff	Colombiano	Cédula	2	17/02/1876	Masculino
Fernando	Vallejo	Mexicano	Cédula	3	24/10/1942	Masculino
Jorge	Franco	Colombiano	Cédula	4	18/05/1977	Masculino

Tabla 1 Datos Tabla Persona

Sobre el archivo de entrada cabe mencionar que para SQL Loader cada línea es un registro, razón por cual es necesario que en el archivo guardado no existan líneas vacías ya que de lo contrario se generará un error. Adicionalmente se debe también asegurar que el separador de las columnas sea efectivamente una coma y no un punto y coma como ocurre al guardar el archivo en formato *.csv* en algunas versiones de Excel.

El archivo final debería verse de la siguiente forma al abrirlo en bloc de notas.

```
Nombre,Apellido,Nacionalidad,Tipo Documento,Número Documento,Fecha Nacimiento,Sexo
Efraín ,Medina,Colombiano,Cédula,1,25/05/1979,Masculino
León,De Grieff,Colombiano,Cédula,2,17/02/1876,Masculino
Fernando,Vallejo,Mexicano,Cédula,3,24/10/1942,Masculino
Jorge,Franco,Colombiano,Cédula,4,18/05/1977,Masculino
```

Preeliminares: Creación del archivo de control

El archivo de control que vamos a utilizar es el siguiente:

```
OPTIONS (SKIP=1)
LOAD DATA
APPEND INTO TABLE Persona
WHEN (nacionalidad='Colombiano')
(
id_persona "cons_persona.nextval",
nombres CHAR TERMINATED BY "," ,
apellidos CHAR TERMINATED BY "," ,
nacionalidad CHAR TERMINATED BY "," ,
tipo_documento CHAR TERMINATED BY "," ,
numero_documento INTEGER EXTERNAL TERMINATED BY ",",
fecha_nacimiento DATE TERMINATED BY "," "to_date(:fecha_nacimiento, 'DD/MM/YYYY')",
sexo CHAR TERMINATED BY "," "case when :sexo='Masculino' then 'm' else 'f' end"
)
```

Se debe copiar las líneas del recuadro a bloc de notas y guardar el archivo en formato *.ctl* (formato por defecto utilizado por SQL Loader para los archivos de control). El archivo mostrado tiene la estructura típica que utilizaremos para un archivo de control, una línea de opciones y la línea *LOAD DATA* seguida de la cláusula *INTO TABLE* y la definición de los campos.

Algunas explicaciones pertinentes propias de este archivo se realizan a continuación. La primera línea *OPTIONS (SKIP=1)* le dice a SQL Loader que ignore el primer registro (i.e. la primera fila) del archivo de datos de origen, en otras palabras estamos diciendo a SQL Loader que ignore el encabezado de los datos.

LOAD DATA le dice a SQL Loader que vamos a insertar datos desde un sistema de archivos a la base de datos. *INTO TABLE* Persona le dice al utilitario que queremos

insertar los datos en la tabla Persona y *APPEND* establece que ello debe hacerse sin borrar los datos preexistentes. Cabe mencionar que existen otros método de cargue diferentes a *APPEND* que borran los datos preexistentes o que generan error si la tabla ya está llena.

Al realizar *WHEN (nacionalidad='Colombiano')* le estamos diciendo a SQL Loader que ignore los registros cuya nacionalidad no sea colombiana, en nuestro caso esto implicará que Fernando Vallejo no será insertado en la tabla (porque no es colombiano y por ende no es persona).

La definición de los campos se encuentra dentro de los paréntesis. Puesto que los campos en el archivo están delimitados por comas, en casi todas la filas aparece la instrucción *TERMINATED BY",".* Observemos de manera detallada como es la definición de cada uno de los campos.

Campo	Definición
id_persona	Este campo corresponde a la llave primaria de la tabla por lo que se recomienda que sea generado mediante una secuencia de números de tal manera que cada registro tenga asignado un número único. Al utilizar el comando *"cons_persona.nextval",* lo que estamos haciendo es asignar un número de la secuencia *cons_persona* a este campo y avanzar dicha secuencia. Por ejemplo, si al comenzar el proceso de cargue el valor de la secuencia es 1 y se va a insertar el primer registro, se asigna el valor 1 al primer registro y se avanza la secuencia de números *cons_persona* a 2.
Nombres Apellidos Nacionalidad Tipo_documento numero_documento	Este caso sólo se está asignando los campos del archivo en orden de aparición a los campos de la tabla. *CHAR | INTEGER EXTERNAL* permite a SQL Loader saber el tipo de datos de lo que se está leyendo. Otros tipos de datos soportados por SQL Loader son *DATE* y *DECIMAL EXTERNAL* cuyo significado es intuitivo
fecha_nacimiento	Además de la definición del campo anteriormente descrita, le estamos diciendo a SQL Loader el formato de la fecha mediante el comando *TO_DATE*: *to_date(:fecha_nacimiento, 'DD/MM/YYYY')*
sexo	Para este campo estamos haciendo una transformación de los archivos de origen: observemos que la definición de la tabla sólo permite ingresar en este campo los valores 'm' y 'f', restricción que los datos de origen no cumplen y que por ende hace necesaria su transformación. Una forma sencilla de realizarlo es mediante la sentencia *CASE* como se aprecia en el ejemplo: *"case when :sexo='Masculino' then 'm' else 'f' end"*

Finalmente podemos mencionar que aunque se utilizaron mayúsculas para las palabras reservadas de SQL Loader, éstas realmente no son sensibles a mayúsculas.

Ejecución de SQL Loader

Ya que tenemos la tabla, el archivo de control y los datos de entrada estamos listos para utilizar la herramienta.

SQL Loader se invoca al igual que SQL Plus desde la consola de comandos de Linux (en nuestro caso de SSH), es decir, desde el paso 2 de *Preeliminares: Creación de la Tabla Persona*. Si en este momento usted se encuentra dentro de SQL Plus, escriba *EXIT* para salir de dicha aplicación y volver a la consola de comandos.

El comando para invocar SQL Loader tiene la forma presentada a continuación:

```
sqlldr NOMBREUSUARIO/CONTRASEÑA@NOMBRESERVICIO control=RUTAARCHIVOCONTROL
log=RUTAARCHIVOLOG data=RUTAORIGENDATOS.
```

RUTAARCHIVOLOG hace referencia a la ubicación donde deseamos que se guarde el archivo de log. El significado de los demás parámetros resulta intuitivo por lo que obviaremos su descripción

Al ejecutar el comando debemos observarse una pantalla similar a:

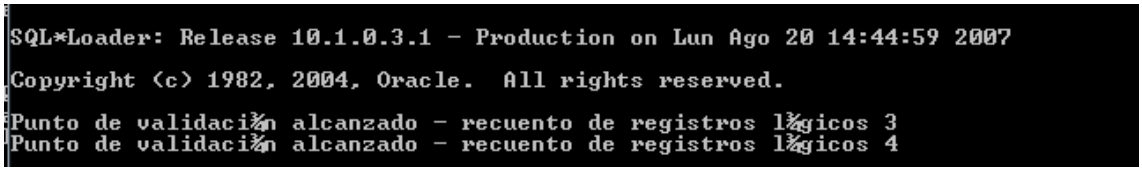

Ilustración 3 Invocación SQL Loader

El archivo de log resultante debe lucir como:

```
 (Permitir todos los registros desechados)

Número a cargar: ALL
Número a ignorar: 1
Errores permitidos: 50
Matriz de enlace:     64 filas, máximo de 256000 bytes
Continuación:    ninguno especificado
Ruta de acceso utilizada:      Convencional

Tabla PERSONA, cargada cuando NACIONALIDAD = 0X436f6c6f6d6269616e6f(carácter 'Colombiano')
Opción INSERT activa para esta tabla: APPEND

   Nombre Columna               Posición   Long  Term Entorno Tipo de dato
------------------------------ ---------- ----- ---- ---- ---------------------
ID_PERSONA                          FIRST     1            CHARACTER
    Cadena SQL para la columna: "cons_persona.nextval"
NOMBRES                              NEXT     *   ,        CHARACTER
APELLIDOS                            NEXT     *   ,        CHARACTER
NACIONALIDAD                         NEXT     *   ,        CHARACTER
TIPO_DOCUMENTO                       NEXT     *   ,        CHARACTER
NUMERO_DOCUMENTO                     NEXT     *   ,        CHARACTER
FECHA_NACIMIENTO                     NEXT     *   ,        DATE DD/MM/RR
    Cadena SQL para la columna: "to_date(:fecha_nacimiento, 'DD/MM/YYYY')"
SEXO                                 NEXT     *   ,        CHARACTER
    Cadena SQL para la columna: "case when :sexo='Masculino' then 'm' else 'f' end"

Registro 3: Desechado - todas las cláusulas WHEN han fallado.

Tabla PERSONA:
  3 Filas se ha cargado correctamente.
  0 Filas no cargada debido a errores de datos.
  1 Fila no cargada porque todas las cláusulas WHEN han fallado.
  0 Filas no cargada porque todos los campos eran nulos.


Espacio asignado a matriz de enlace:           115840 bytes (64 filas)
Bytes de buffer de lectura: 1048576

Total de registros lógicos ignorados:          1
Total de registros lógicos leídos:           4
Total de registros lógicos rechazados:         0
Total de registros lógicos desechados:         1

La ejecución empezó en Lun Ago 20 14:46:24 2007
La ejecución terminó en Lun Ago 20 14:46:24 2007

Tiempo transcurrido:     00:00:00.08
Tiempo de CPU:         00:00:00.04
```

Ilustración 4 Archivo de Log

Carga Selectiva de Registros y Columnas

En ocasiones hay datos que no se quiere que sean insertados, razón por la cual se deben añadir condiciones a los campos mediante la cláusula *WHEN*. Dicha cláusula se inserta justo debajo de la cláusula *INTO TABLE* y su uso se puede describir de manera simplificada mediante la sintaxis:

```
WHEN (condition AND condition AND condition AND condition…)
condition := fieldname = | <> | != 'string'  (e.g.  nombre <> 'Julián')
```

Nótese que no es posible unir las condiciones mediante el operador *OR*, sólo se permite *AND*. Ejemplo de un archivo de control con cláusula *WHEN* es:

```
LOAD DATA
APPEND INTO TABLE Medico
WHEN (profesion='medico') AND (pais='Colombia')
(
cedula CHAR TERMINATED BY ', ,
nombre CHAR TERMINATED BY ',',
apellido CHAR TERMINATED BY ','',
)
```

Por otra parte, si lo que se quiere es no llenar un atributo la forma más simple es mediante la cláusula *FILLER*. El siguiente ejemplo muestra su utilización.

Supongamos que tenemos los datos:

```
123, Manzana,Fruta
999,Perro,Animal
666,Gato,Animal
```

Que corresponden, al código, el nombre y la descripción de COSAS. Si deseamos obviar el primer campo el archivo de control sería:

```
LOAD DATA
APPEND INTO TABLE Cosas
 (
codigo FILLER CHAR TERMINATED BY ',' ,
nombre  CHAR TERMINATED BY ',' ,,
descripcion CHAR TERMINATED BY ','
)
```

SQL Loader no cargará el código de las cosas, únicamente el nombre y la descripción.

Carga de Múltiples Tablas

Una forma de cargar múltiples tablas utilizando un archivo de datos para cada tabla se puede lograr aplicando las cláusulas *WHEN, INTO TABLE* e *INFILE*.

Supongamos que queremos llenar las tablas PROFESION y PROYECTO definidas en las siguientes sentencias SQL.

```
CREATE TABLE  profesion
(id_profesion  NUMBER CONSTRAINT profesion_pk PRIMARY KEY,
nombre  VARCHAR2(30) CONSTRAINT profesion_nombre_nn NOT NULL CONSTRAINT profesion_nombre_nd
UNIQUE,
descripcion VARCHAR2(60)
) ;

CREATE SEQUENCE cons_profesion;

CREATE TABLE  proyecto
(id_proyecto  NUMBER CONSTRAINT proyecto_pk PRIMARY KEY,
nombre  VARCHAR2(30) CONSTRAINT proyecto_nombre_nn NOT NULL CONSTRAINT proyecto_nombre_nd
UNIQUE,
precio_estimado NUMBER  CONSTRAINT proyecto_precio_nn NOT NULL,
descripcion VARCHAR2(60)
) ;

CREATE SEQUENCE cons_proyecto;
```

El archivo fuente para PROFESION tiene la siguiente forma:

```
profesión,médico,cura personas
profesión,veterinario,cura animales
profesión,arquitecto,construye edificios
```

Y para PROYECTO:

```
proyecto,palacio Rosado,27000,construcción de un palacio
proyecto,avenida Primera,13000,construcción una avenida
proyecto,hospital San Juan,16500,hospital de beneficencia
```

El archivo de control que carga los datos es:

```
LOAD DATA
INFILE 'profesion.csv'
INFILE 'proyecto.csv'
INTO TABLE profesion
TRUNCATE
WHEN (tipo_tabla ='profesión')
(
tipo_tabla FILLER CHAR TERMINATED BY ',' ,
id_profesion "cons_profesion.nextval",
nombre CHAR TERMINATED BY ',' ,
descripcion CHAR TERMINATED BY ','
)
INTO TABLE proyecto
REPLACE
WHEN (tipo_tabla ='proyecto')
(
tipo_tabla FILLER POSITION(1) CHAR TERMINATED BY ',' ,
id_proyecto "cons_proyecto.nextval",
nombre CHAR TERMINATED BY ',' ,
precio_estimado INTEGER EXTERNAL TERMINATED BY ',' ,
descripcion CHAR TERMINATED BY ','
)
```

En las líneas 2 y 3 del archivo se utilizó *INFILE* para seleccionar los archivos fuente de cada tabla (con lo que ya no es necesario el parámetro *data* en la línea de comandos), la ruta de los archivos es una ruta relativa por lo que es necesario que estén en el mismo directorio del archivo de control.

La cláusula *WHEN* es necesaria puesto que SQL Loader ve a los dos archivos como un solo archivo lógico, *WHEN* es lo que determina a que tabla pertenece cada registro. *FILLER* se utilizó para ignorar la primera columna al momento de llenar las tablas, es decir la columna que indica a que tabla pertenece el registro. *POSITION(1)* se empleó para devolver a SQL Loader al principio del registro en el archivo de datos, de tal forma que cada registro vuelva a ser analizado con el propósito de determinar si pertenece a la segunda tabla.

Finalmente notemos la variante utilizada en la cláusula *INTO TABLE*:

- A diferencia de ejemplos anteriores en los que usábamos *APPEND* para llenar las tablas, en esta ocasión usamos *TRUNCATE* y *REPLACE.* Éstas variantes borran los datos preexistentes en las tablas (la diferencia entre los dos es que con *TRUNCATE* no es posible hacer rollback).
- El método de inserción de los datos, entiéndase *TRUNCATE* | *REPLACE* | *APPEND* va antecedido del nombre de la tabla y no seguido de *INTO TABLE*, ello hace que el alcance del método de carga sea local y sea posible especificar uno para cada tabla.
-

```
LOAD DATA
APPEND
INTO TABLE tabla1
(
…
)
INTO TABLE tabla2
(
…
)
```

Global

```
LOAD DATA
INTO TABLE tabla1 APPEND
(
…
)
INTO TABLE tabla2 REPLACE
(
…
)
```

Local

Transformaciones a los Datos

En SQL Loader es posible introducir una secuencia SQL al final de la definición de un campo para realizar transformaciones a los datos.

Un ejemplo de un archivo de control que realiza múltiples transformaciones se presenta a continuación.

```
LOAD DATA
  INFILE *
  INTO TABLE modified_data
  ( rec_no                "my_db_sequence.nextval",
    time_loaded           "to_char(SYSDATE, 'HH24:MI')",
    data1       TERMINATED BY “,” ":data1/100",
    data2       TERMINATED BY “,”  "upper(:data2)",
    data3       “to_date(:data1, 'YYMMDD')”
  )
  BEGINDATA
  1111111,aaaaBa
  2211222,9901112
```

Cuatro puntos se destacan del ejemplo, el primero es que las sentencias o comandos SQL utilizados en la definición del campo deben encerrarse entre comillas dobles. El segundo es que para referirse a un campo dentro de la sentencia se utiliza la sintaxis *:nombreCampo.* El tercero es que las sentencias no sólo pueden referirse al campo que se está definiendo, es posible referirse a otro campo, por ejemplo el campo data3 se crea al convertir el valor del campo data1 en una fecha.

Referencias

[1] JONATHAN GENNICK, SANJAY MISHRA. Oracle Sql Loader The Definitive Guide. Oreilly 2001.