SQL Loader Manual en español para principiantes

diciembre 4, 2017 by mariocelis

SQL Loader es una de las tecnologías de ORACLE más utilizadas para importar información a las bases de datos de ORACLE.

Esta tecnología consiste en tener información valiosa en archivos planos, por ejemplo: se puede cargar información de clientes, facturas, usuarios, notas de crédito, notas de débito, cuentas bancarias, etc.

Los archivos que se van a subir mediste **SQL Loader** tienen que tener un formato especifico, ya sea por posiciones o que se utilice un separador para reconocer el inicio y fin de cada dato dentro del archivo.

SQL Loader toma la información del archivo plano y la procesa, de este proceso se pueden generas varios archivos mas, como son:

Un archivo de LOG: el cual va a contener un informe de lo que se cargo con éxito y de lo que fallo en la carga, este archivo es muy util para realizar análisis y correcciones.

El archivo de errores: Este archivo contiene todos los registros que por algún motivo no se pudieron cargar a la base de datos.

A continuación se muestra un diagrama que **muestra el flujo del funcionamiento de carga de información con SQL Loader**

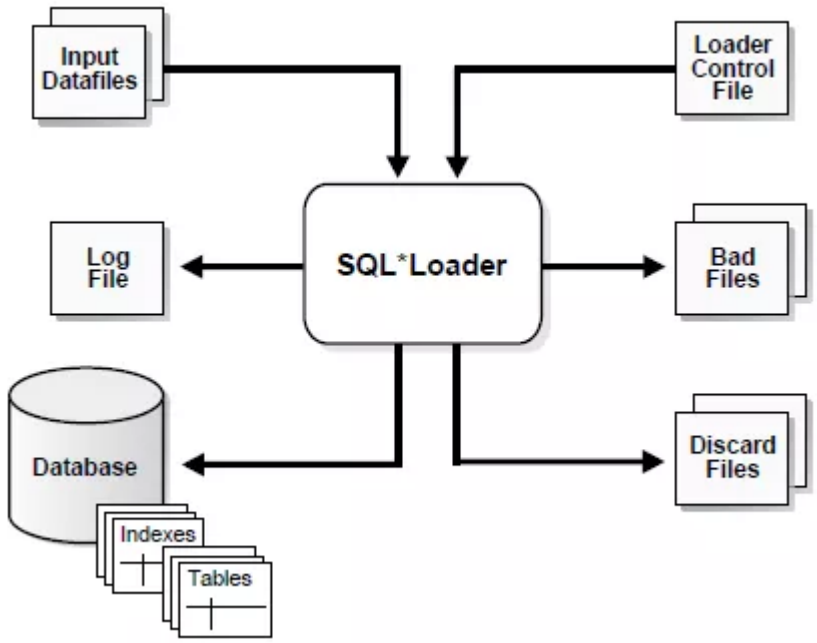
Tabla de contenido

1. SQL Loader
    1.1. Archivo de control (CTL):
    1.2. Conversión de datos y especificación de Datatype
    1.3. Registros descartados y rechazados
    1.4. Archivo de descartados
    1.5. Archivo LOG
    1.6. Invocando SQL Loader
    1.7. Control file contents
    1.8. Especificando DATAFILE
    1.9. Specifying Multiple Datafiles

1.15. Cargar registros basados en condiciones
1.16. Especificando delimitadores (Default data)
1.17. Extracting Multiple Logical Records
1.17.1. Posición relativa basada en delimitadores
1.17.2. Distinguiendo diferentes formatos de registro de entrada
1.17.3. Posición relativa basada en el parámetro POSITION
1.18. Contenido de la lista de campos
1.19. Publicaciones relacionadas

SQL Loader

Diagrama de descripción general.



Archivo de control (CTL):

Es un archivo de texto escrito en un lenguaje que **SQL Loader** entiende. El archivo de control le dice a **SQL Loader** donde encontrar la información, como analizar e interpretar la información, donde se insertara y más.

Se puede decir que un CTL tiene tres secciones:

La 1ra tiene información de toda la sesión, por ejemplo:

La 2da sección consiste de uno o más bloques INTO TABLE. Cada uno de estos bloques contiene información acerca de la tabla dentro de la cual se cargara la información, como son el nombre de la tabla y las columnas.

La 3ra sección es opcional y, si se presenta, contiene información de entrada.

Información de entrada y Archivos de datos

SQL *Loader lee información de uno o más archivos especificados en el archivo de control.

Conversión de datos y especificación de Datatype

Durante una ruta de carga, los campos de datos en el archivo de datos son convertidos dentro de columnas en la BD.

1. SQL Loader usa la especificación de campos en el archivo de control para interpretar el formato del datafile, analiza los datos de entrada, y pobla el bind array correspondiente a la sentencia SQL INSERT usando la información.
2. La BD ORACLE acepta la información y ejecuta la sentencia INSERT para almacenar la información en la BD.

La BD ORACLE usa los datatype de la columna para convertir la información en su forma final.

Registros descartados y rechazados

Los registros leídos desde el archivo de entrada podrían no ser insertados en la BD. Cada registro podría ser colocado en un archivo **bad** o **discart**.

Bad file:

El archivo bad tiene registros que fueron rechazados, si no especificas un bad file y hay registros rechazados, SQL Loader crea uno automáticamente. Tiene el mismo nombre del archivo de datos pero con extensión .bad.

SQL Loader Rejects

Los registros son rechazados cuando el formato e entrada es inválido.

criterio de selección de registros especificada en el archivo de control.

Archivo LOG

Cuando **SQL Loader** inicia la ejecución, crea un log file. El log file contiene un resumen detallado de la carga, incluyendo una descripción de algunos errores ocurridos durante la carga.

Invocando SQL Loader

Cuando se invoca **SQL Loader** puedes especificar ciertos parámetros para establecer las características de la sesión. Los parámetros pueden ser introducidos en cualquier orden, separados por comas opcional mente. Tu especificas los valores para los parámetros, o en algunos casos, puedes aceptar el default sin introducir un valor.

Ejemplo:

SQLLDR CONTROL=sample.ctl, LOG=sample.log, BAD=baz.bad, DATA=etc.dat USERID=scott/tiger, ERRORS=999, LOAD=2000, DISCARD=toss.dsc,

DISCARDMAX=5

Si ejecutas el comando de SQL *Loader sin parámetros se muestra su pantalla de ayuda.

Códigos de salida:

SQL *Loader provee los resultados de la ejecución después de terminada.

All or some rows discarded	EX_WARN
Discontinued load	EX_WARN
Command-line or syntax errors	EX_FAIL
Oracle errors nonrecoverable for SQL*Loader	EX_FAIL
Operating system errors (such as file open/close and malloc)	EX_FAIL

For UNIX, the exit codes are as follows:

```
EX_SUCC 0
EX_FAIL 1
EX_WARN 2
EX_FTL  3
```

For Windows NT, the exit codes are as follows:

```
EX_SUCC 0
EX_WARN 2
EX_FAIL 3
EX_FTL  4
```

Se puede hacer un chequeo con un Shell script como sigue:

```
#!/bin/sh
sqlldr scott/tiger control=ulcase1.ctl log=ulcase1.log
retcode=`echo $?`
case "$retcode" in
0) echo "SQL*Loader execution successful" ;;
1) echo "SQL*Loader execution exited with EX_FAIL, see logfile" ;;
2) echo "SQL*Loader execution exited with EX_WARN, see logfile" ;;
3) echo "SQL*Loader execution encountered a fatal error" ;;
*) echo "unknown return code";;
esac
```

Control file contents

El archivo de control es un archivo de texto que contiene lenguaje de definición de datos (DDL).

En general, el archivo de control tiene tres secciones principales en el siguiente orden:

A) Información de sesión.

B) Información de tablas y campos.

```
2    LOAD DATA
3    INFILE 'sample.dat'
4    BADFILE 'sample.bad'
5    DISCARDFILE 'sample.dsc'
6    APPEND
7    INTO TABLE emp
8    WHEN (57) = '.'
9    TRAILING NULLCOLS
10   (hiredate SYSDATE,
      deptno POSITION(1:2)  INTEGER EXTERNAL(2)
             NULLIF deptno=BLANKS,
       job   POSITION(7:14)  CHAR  TERMINATED BY WHITESPACE
             NULLIF job=BLANKS  "UPPER(:job)",
       mgr   POSITION(28:31) INTEGER EXTERNAL
             TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
       ename POSITION(34:41) CHAR
             TERMINATED BY WHITESPACE  "UPPER(:ename)",
       empno POSITION(45) INTEGER EXTERNAL
             TERMINATED BY WHITESPACE,
       sal   POSITION(51) CHAR  TERMINATED BY WHITESPACE
             "TO_NUMBER(:sal,'$99,999.99')",
       comm  INTEGER EXTERNAL  ENCLOSED BY '(' AND '%'
             ":comm * 100"
    )
```

Descripción de cada número del ejemplo anterior.

1. Es un comentario introducido en el archivo de control.
2. LOAD DATA: Llama a SQL Loader que indica que es el inicio de una nueva carga.
3. INFILE: Indica el nombre del archivo que contiene la información que se quiere cargar.
4. BADFILE: Indica el nombre del archivo dentro del cual los registros rechazados son colocados.
5. DISCARDFILE: Indica el nombre del archivo donde los registros descartados son colocados.
6. APPEND: Es una de las opciones que puedes usar cuando cargas información dentro de una tabla que no está vacía. Para cargar información dentro de una tabla que esta vacía, debes usar la cláusula INSERT.
7. INTO TABLA: Permite identificar las tablas, campos y tipos de datos. Esto define relaciones entre registros en el datafile en tablas en la BD.
8. WHEN: Indica uno o más campos de condición. SQL *Loader decide si/no cargar registros en base a la condición del campo.
9. TRAILING NULLCOLS: Establece alguna columna posicionada que no está presente en el registro como NULL.
10. El resto del archivo de control contiene la lista de campos, los cuales proveen información acerca de los formatos de columnas en la tabla que inicia la carga.

Comentarios dentro del archivo de control:

Especificando DATAFILE

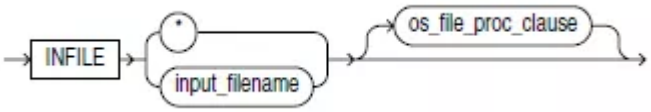
Para especificar los archivos de datos que contienen la información a cargar use la palabra clave INFILE, seguida por el nombre del archivo y opcionalmente (option string). Puedes especificar múltiples archivos usando múltiples INFILE.

Nota: puedes especificar el archivo de datos desde la línea de comandos, usando el parámetro DATA. Un nombre de archivo especificado en línea de comandos sobrescribe el primer INFILE en el archivo de control.

Si no se especifica un nombre de archivo, el nombre de archivo por default pasa con extensión .dat.

Si el archivo de control contiene la información a cargar, especifique un (*).

Sintaxis:



Ejemplos:

The following list shows different ways you can specify `INFILE` syntax:

- Data contained in the control file itself:

```
INFILE *
```

- Data contained in a file named `sample` with a default extension of `.dat`:

```
INFILE sample
```

- Data contained in a file named `datafile.dat` with a full path specified:

```
INFILE 'c:/topdir/subdir/datafile.dat'
```

Nota: Los nombres de archivo que incluyen espacios o puntuaciones deben ser encerradas en comillas simples.

Specifying Multiple Datafiles

También puedes especificar separadamente un discart file y un bad file para cada datafile. En cada caso, los archivos bad y discart se deben declarar inmediatamente después del nombre de cada nombre del datafile.

Ejemplos:

```
INFILE  mydat1.dat  BADFILE  mydat1.bad  DISCARDFILE mydat1.dis
INFILE  mydat2.dat
INFILE  mydat3.dat  DISCARDFILE  mydat3.dis
INFILE  mydat4.dat  DISCARDMAX  10 0
```

Identificando información e el archivo de control con BEGINDATA

Si la información es incluida en el mismo archivo de control, entonces la cláusula INFILE es seguida por un (*).

Especificando la sentencia BEGINDATA antes del primer registro de datos.

Sintaxis:

BEGINDATA

data

Si omites el BEGINDATA pero incluyes información en el CTL, SQL *Loader intenta interpretar la información como información de control y lanza un mensaje de error. Si la información está en un archivo separado, no use BEGINDATA.

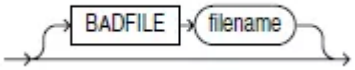
No use espacios o otros caracteres en la misma línea del BEGINDATA, o la línea del BEGINDATA será interpretada como la primera línea de información.

No ponga comentarios después del BEGINDATA, o ellos serán interpretados como información.

Especificando el BAD FILE

Cuando se ejecuta SQL Loader, este puede crear un archivo llamado (bad file) o (reject file) en el cual son colocados los registros que son rechazados por errores de formato por errores de ORACLE.

Sintaxis:



Ejemplos:

A) BADFILE simple

B) BADFILE bad0001.rej

C) BADFILE '/REJECT_DIR/bad0001.rej'

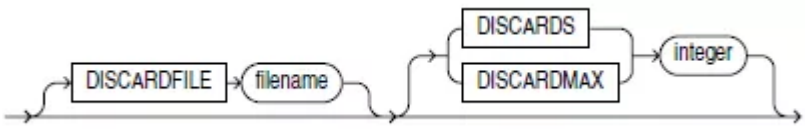
Un registro puede ser rechazado por las siguientes razones:

1. Una inserción, el registro causa un error (como un dato invalido para un tipo e dato dado).
2. El registro es formado incorrectamente, SQL Loader no encuentra los límites del registro.
3. El registro viola una restricción.

Especificando el Discart File

Durante la ejecución, SQL *Loader puede crear un (discart file) para los registros que no satisfacen algún criterio de carga. Los registros contenidos en este archivo son llamados descartados. Los registros descartados no satisfacen alguna clausula WHEN especificada en el CTL. Estos registros difieren de los rechazados. Los registros descartados no necesariamente tienen información mal.

Sintaxis:



Ejemplos:

A) DISCARDFILE circular

Parametro CHARACTERSET

Especificando el parámetro CHARACTERSET SQL Loader llama al juego de caracteres del archive de entrada. El juego de caracteres por defecto para todos los datafiles, sino se especifica el CHARACTERSET, es el definido en NLS_LANG.

Sintaxis:

CHARACTERSET char_set_name

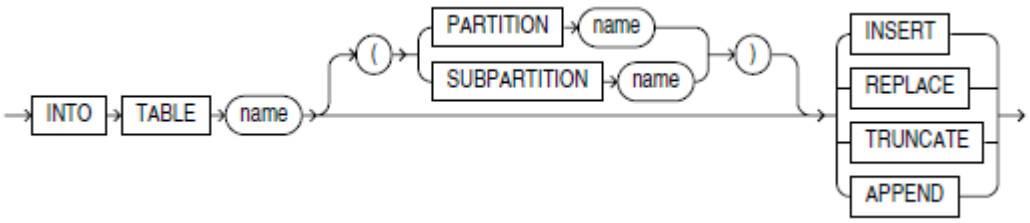
Especificando el nombre de tablas

La cláusula INTO TABLE de la sentencia LOAD DATA permite que identifiques tablas, campos y tipos de datos.

Clausula INTO TABLE:

Para cargar multiples tablas, incluya un INTO TABLE para cada tabla que quieras cargar.

Sintaxis:



Ejemplos:

```
INTO TABLE scott."CONSTANT"
INTO TABLE scott."Constant"
INTO TABLE scott."-CONSTANT"
```

Métodos de carga

Cuando cargas una tabla puedes usar la cláusula INTO TABLE para especificar un método de carga (INSERT, APPEND, REPLACE y TRUNCATE), este se debe especificar antes de la cláusula INTO TABLE.

Carga de datos en tablas que no están vacías

APPEND: Si existe información en la tabla, se agregan nuevos registros a ella si no existe información, simplemente se cargan los nuevos registros.

REPLACE: Todos los registros en la tabla son eliminados y la nueva información es cargada, debe tener el privilegio DELETE.

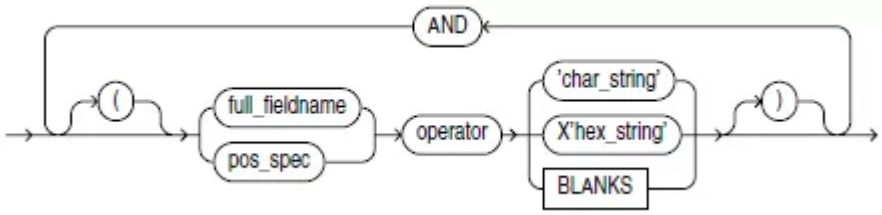
TRUNCATE: Elimina rápida y eficientemente todos los registros de la tabla, para un mejor rendimiento.

Precaución: cuando especifique REPLACE o TRUNCATE, la tabla entera es reemplazada, no renglones individuales, después de que son borrados se realiza un commit.

Cargar registros basados en condiciones

Puedes decidir cargar o descartar un registro lógico usando la cláusula WHEN. La cláusula WHEN aparece después del nombre de la tabla y es seguida por una o más condiciones.

Sintaxis:



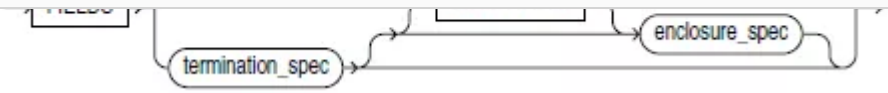
Ejemplos:

```
WHEN (5) = 'q'
```

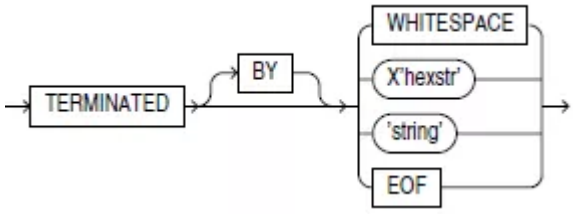
```
WHEN (deptno = '10') AND (job = 'SALES')
```

Especificando delimitadores (Default data)

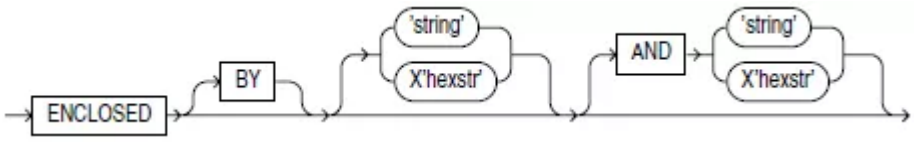
Si todos los campos terminan similarmente en el datafile, puedes usar la cláusula FIELDS para indicar los delimitadores por defecto.



Termination_spec

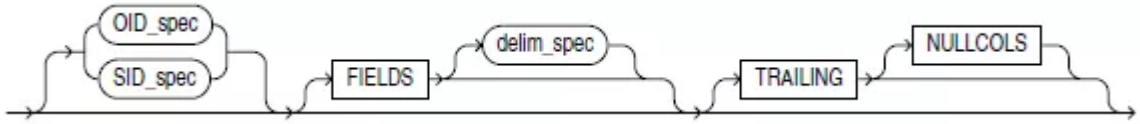


Enclosure_spec



Manejo de registros con datos faltantes:

Cuando un CTL especifica más campos para un registro, SQL Loader especifica las columnas que deben ser consideradas como NULL o se generara un error. SQL Loader usa la presencia o ausencia de TRAILING NULLCOLS (para determinar el curso de la acción).



Clausula TRAILING NULLCOLS:

Esta cláusula trata las columnas de posición relativa que no están presentes en el registro como columnas NULL.

Ejemplo:

```
  dname  CHAR TERMINATED BY WHITESPACE,
  loc    CHAR TERMINATED BY WHITESPACE
)
```

En este caso, el campo “loc” se establece a NULL, sin la cláusula TRAILING NULLCOLS se generaría un error debido a la falta de datos.

Beneficios de usar múltiples clausulas INTO TABLE:

1. Cargar información dentro de diferentes tablas
2. Extraer múltiples registros lógicos desde un simple registro de entrada
3. Distinguir diferentes formatos de registro de entrada

Extracting Multiple Logical Records

SQL Loader trata un simple registro físico en el archivo de entrada como dos registros lógicos y usa dos clausulas INTO TABLE para cargar la información en la tabla “emp”

Datos:

```
1119 Smith      1120 Yvonne
1121 Albert     1130 Thomas
```

CTL:

```
INTO TABLE emp
     (empno POSITION(1:4)  INTEGER EXTERNAL,
      ename POSITION(6:15) CHAR)
INTO TABLE emp
     (empno POSITION(17:20) INTEGER EXTERNAL,
      ename POSITION(21:30) CHAR)
```

Posición relativa basada en delimitadores

El mismo registro debe ser cargado con diferente especificación. El siguiente CTL usa posicionamiento relativo. Esto especifica que cada campo es delimitado por un simple espacio en blanco (“ ”) o con un número indeterminado de blancos y tabs (WHITESPACE):

El punto importante en este ejemplo es que el segundo campo “empno” es encontrado inmediatamente después del primer “ename”, el escaneo de campo no inicia sobre el inicio del registro para una nueva clausula INTO TABLE, el escaneo continua donde se quedó.

Para forzar que el escaneo de registro inicie en un lugar específico, use l parámetro POSITION.

Distinguiendo diferentes formatos de registro de entrada

Un simple datafile puede contener registros en diferentes formatos. Considere los siguientes datos, en los cuales hay registros intermedios “emp” y “dep”.

```
1 50   Manufacturing       – DEPT record
2 1119 Smith      50       – EMP record
2 1120 Snyder     50
1 60   Shipping
2 1121 Stevens    60
```

Un campo record ID distingue entre los dos formatos. Los registros de departamentos tienen un 1 en la primera columna, los registros de empleados tienen un 2. Ejemplo:

```
INTO TABLE dept
   WHEN recid = 1
   (recid  FILLER POSITION(1:1)  INTEGER EXTERNAL,
    deptno POSITION(3:4)  INTEGER EXTERNAL,
    dname  POSITION(8:21) CHAR)
INTO TABLE emp
   WHEN recid <> 1
   (recid  FILLER POSITION(1:1)   INTEGER EXTERNAL,
    empno  POSITION(3:6)   INTEGER EXTERNAL,
    ename  POSITION(8:17)  CHAR,
    deptno POSITION(19:20) INTEGER EXTERNAL)
```

Posición relativa basada en el parámetro POSITION

Aquí es necesario usar el parámetro POSITION.

```
    WHEN recid <> 1
    (recid  FILLER POSITION(1) INTEGER EXTERNAL TERMINATED BY ' ',
     empno  INTEGER EXTERNAL TERMINATED BY ' '

ename  CHAR TERMINATED BY WHITESPACE,
deptno INTEGER EXTERNAL TERMINATED BY ' ')
```

El parámetro **POSITION** en la segunda clausula INTO TABLE es necesario para cargar la información correctamente. Esto causa que el registro se escanee al inicio sobre la columna 1 cuando se cheque que la información coincida en el segundo formato.

Campos que pueden variar en tamaño de renglón a renglón son:

- CHAR
- DATE
- INTERVAL DAY TO SECOND
- INTERVAL DAY TO YEAR
- LONG VARRAW
- numeric EXTERNAL
- TIME
- TIMESTAMP
- TIME WITH TIME ZONE
- TIMESTAMP WITH TIME ZONE
- VARCHAR
- VARCHARC
- VARGRAPHIC
- VARRAW
- VARRAWC

Cuando los datatypes (CHAR, DATE, y NUMERIC EXTERNAL) son especificados con delimitadores, alguna longitud especificada para estos campos es la máxima longitud. Cuando se especifica in delimitadores, el tamaño en el registro es fijo.

Contenido de la lista de campos

```
.
1  (hiredate  SYSDATE,
2     deptno  POSITION(1:2)  INTEGER EXTERNAL(2)
              NULLIF deptno=BLANKS,
3       job   POSITION(7:14)  CHAR  TERMINATED BY WHITESPACE
              NULLIF job=BLANKS  "UPPER(:job)",
        mgr   POSITION(28:31) INTEGER EXTERNAL
              TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
        ename POSITION(34:41) CHAR


              TERMINATED BY WHITESPACE  "UPPER(:ename)",
        empno POSITION(45) INTEGER EXTERNAL
              TERMINATED BY WHITESPACE,
        sal   POSITION(51) CHAR  TERMINATED BY WHITESPACE
              "TO_NUMBER(:sal,'$99,999.99')",
4       comm  INTEGER EXTERNAL  ENCLOSED BY '(' AND '%'
              ":comm * 100"
    )
```

Descripción de los números del 1…4 anteriores:

1. SYSDATE establece a la columna la fecha actual del sistema.
2. POSITION especifica la posición de un datafield
   INTEGER EXTERNAL es el tipo de dato del campo
   NULLIF es una de las cláusulas que pueden ser usadas para especificar condiciones.
   En este ejemplo, el campo se inicia comparando con blancos, usando el parámetro BLANKS.
3. TERMINATED BY WHITESPACE es uno de los delimitadores que es posible especificar para un campo.
4. ENCLOSED BY es otro posible delimitador de campos.

Espero que esta documentación te sirva, a continuación te dejo unos links donde puedes consultar mas.

Enlaces Internos

Tutorial de Java SE desde cero en español

Documentación de SQL *Loader