达梦数据库

达梦技术手册

DM8 快速装载工具使用手册

Service manual of DM8 fast loader

前言

概述

本文档主要介绍如何使用 DM8 快速装载工具进行 DM 数据库中表数据的快速载入和载出。

读者对象

本文档主要适用于 DM 数据库的：

- 开发工程师
- 测试工程师
- 技术支持工程师
- 数据库管理员

通用约定

在本文档中可能出现下列标志，它们所代表的含义如下：

表 0.1 标志含义

标志	说明
 警告：	表示可能导致系统损坏、数据丢失或不可预知的结果。
 注意：	表示可能导致性能降低、服务不可用。
 小窍门：	可以帮助您解决某个问题或节省您的时间。
 说明：	表示正文的附加信息，是对正文的强调和补充。

在本文档中可能出现下列格式，它们所代表的含义如下：

表 0.2 格式含义

格式	说明
宋体	表示正文。
黑体	标题、警告、注意、小窍门、说明等内容均采用黑体。
Courier new	表示代码或者屏幕显示内容。
粗体	表示命令行中的关键字（命令中保持不变、必须照输的部分）或者正文中强调的内容。
<>	语法符号中，表示一个语法对象。
::=	语法符号中，表示定义符，用来定义一个语法对象。定义符左边为语法对象，右边为相应的语法描述。
|	语法符号中，表示或者符，限定的语法选项在实际语句中只能出现一个。
{ }	语法符号中，大括号内的语法选项在实际的语句中可以出现 0…N 次(N 为大于 0 的自然数)，但是大括号本身不能出现在语句中。
[ ]	语法符号中，中括号内的语法选项在实际的语句中可以出现 0…1 次，但是中括号本身不能出现在语句中。
关键字	关键字在 DM_SQL 语言中具有特殊意义，在 SQL 语法描述中，关键字以大写形式出现。但在实际书写 SQL 语句时，关键字既可以大写也可以小写。

访问相关文档

如果您安装了 DM 数据库，可在安装目录的“\doc”子目录中找到 DM 数据库的各种手册与技术丛书。

您也可以通过访问我们的网站阅读或下载 DM 的各种相关文档。

联系我们

如果您有任何疑问或是想了解达梦数据库的最新动态消息，请联系我们：

网址：www.dameng.com

技术服务电话：400-991-6599

技术服务邮箱：dmtech@dameng.com

目录

1 概述

1.1 功能简介

DM 提供了两种形式的快速装载工具：一是 dmfldr；二是 dmldrc 和 dmldrp。用户通过使用快速装载工具能够把按照一定格式排序的文本数据以简单、快速、高效的方式载入到 DM 数据库中，或把 DM 数据库中的数据按照一定格式载出到文本文件中。

两种形式的工具，功能完全一样，区别是应用场景不同。在软硬件资源充裕的情景下，首选 dmfldr 工具，dmfldr 在一台机器上启动即可，独立完成快速装载任务，简单高效。在机器资源匮乏的极端情景下，需选择 dmldrc 和 dmldrp 工具，分别部署在两台机器上，各自占用较少的机器资源，相互配合共同完成快速装载任务，可克服机器资源不足的问题。

其中，表及表的同义词支持数据载入和载出，视图及视图的同义词仅支持数据载出。

1.2 系统结构

1.2.1dmfldr 结构

dmfldr（DM Fast Loader）包含 dmfldr 客户端和 dmfldr 模块两部分。dmfldr 客户端实现初始化快速装载环境、读取数据、打包数据和发送数据功能。dmfldr 功能模块嵌入在数据库服务器中，实现装载功能。两者相互协作，共同完成 dmfldr 的各项功能。

dmfldr 的系统结构如图 1.1 所示。

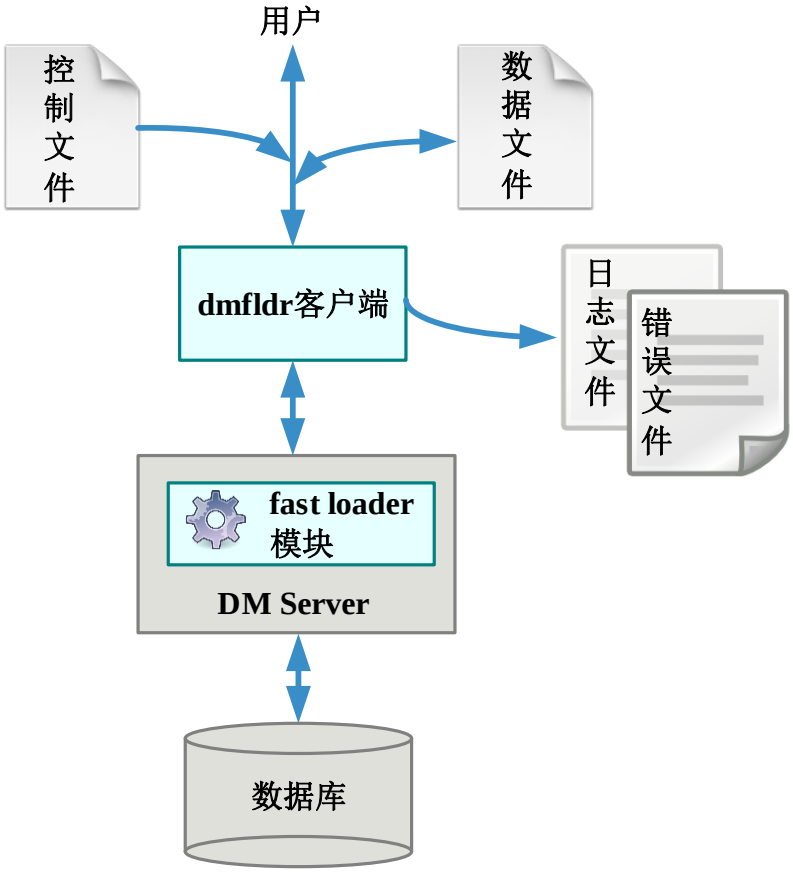

图 1.1 dmfldr 结构示意图

当进行数据载入时，dmfldr 客户端接收用户提交的命令与参数，分析控制文件与数据文件，将数据打包发送给服务器端的 dmfldr 模块，由 dmfldr 模块完成数据的真正装载工作。并分析服务器返回的消息，必要时根据用户参数指定生成日志文件与错误数据文件。

当进行数据载出时，dmfldr 客户端接收用户提交的命令与参数，分析控制文件，将用户要求转换成相应消息发送给服务器端的 dmfldr 模块。dmfldr 模块解析并打包需要导出的数据，发送给 dmfldr 客户端，客户端将数据写入指定的数据文件，必要时根据用户参数指定生成日志文件。

1.2.2dmldrc 和 dmldrp 结构

快速装载工具所在机器的软硬件资源匮乏会影响快速装载的性能。为了应对这种极端情况，DM 提供了一种轻量型的快速装载工具套装：dmldrc 和 dmldrp。dmldrc 为轻量级快速装载工具的客户端，负责初始化快速装载环境和读取数据；dmldrp 为轻量级快速装载工具的服务器，负责打包数据和发送数据。两者相互配合完成和 dmfldr 客户端一样的

功能。dmldrc 和 dmldrp 可部署在不同的机器上。将快速装载工具的任务分配到两个独立的工具上，并部署在不同的机器上，可减轻机器的软硬件压力。

用户还可以根据需要，配置多个 dmldrc 同时工作。dmldrp 为每一个 dmldrc 分配一个专属子进程，专门处理该 dmldrc 读取的数据。多个 dmldrc 相互独立，互不影响。

其它模块和 dmfldr 工具中的模块功能一样。

dmldrc 和 dmldrp 工具套装的系统结构如图 1.2 所示。

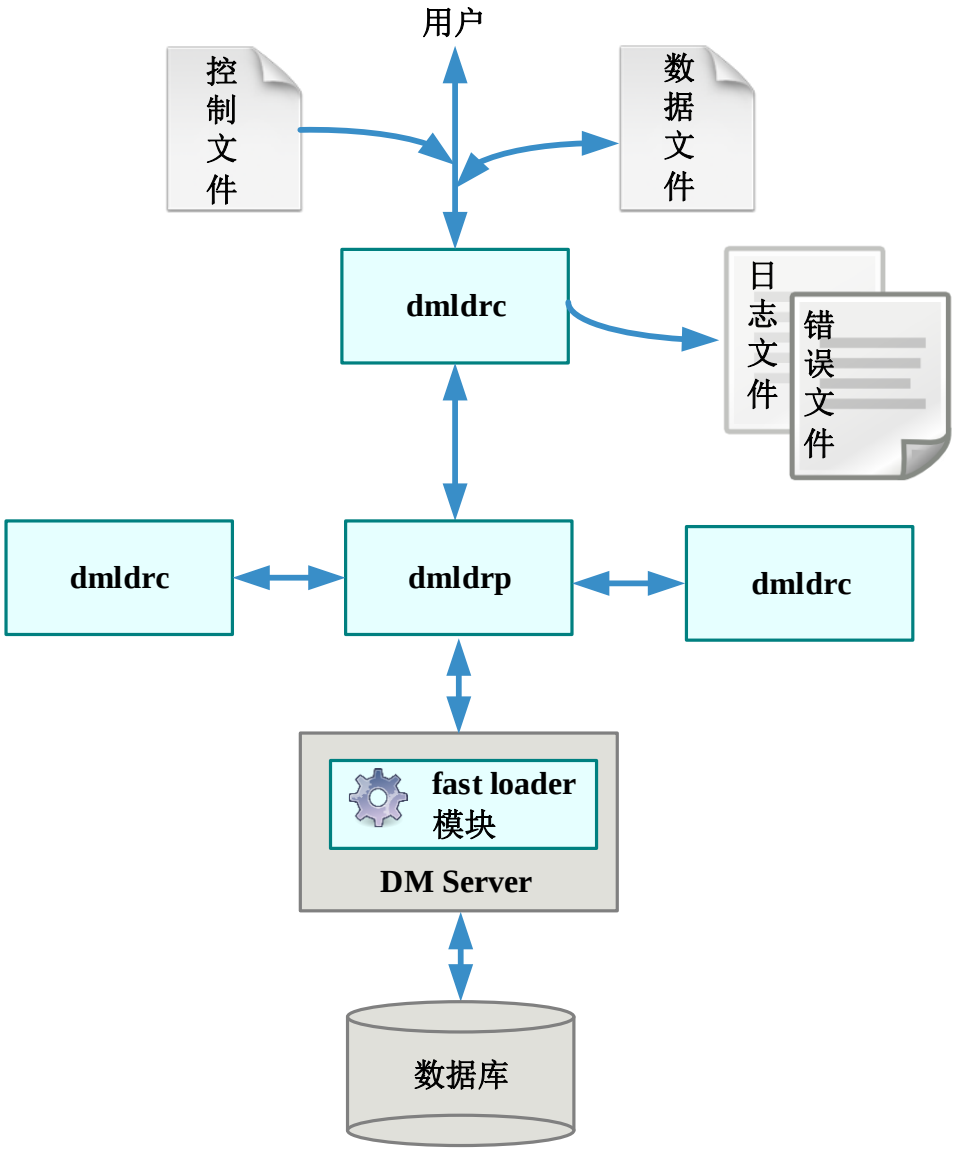

图 1.2 dmldrc 和 dmldrp 结构示意图

2 dmfldr 入门

本章简单介绍如何启动 dmfldr 和 dmfldr 支持的参数简介。通过阅读本章，读者可以了解 dmfldr 通过各参数能提供的各项功能，不过要想熟练灵活地使用 dmfldr 还需要继续阅读下一章。

2.1 启动 dmfldr

安装好 DM 数据库管理系统后，在安装目录的“bin”子目录下可找到 dmfldr 执行文件。

启动操作系统的命令行窗口，进入“dmfldr”所在目录，可以准备启动 dmfldr 工具了。

dmfldr 的使用必须指定必要的参数，否则工具会报错“无效的参数个数”并退出。为 dmfldr 指定参数的格式为：

```
dmfldr keyword=value [keyword=value ...]
```

USERID 是启动 dmfldr 必须要指定的参数，且 USERID 必须是第一个参数。其它参数用户需根据实际情况选取。

例 1 使用 USERID 和 CONTROL 参数启动 dmfldr，完成载入。

```
dmfldr USERID=SYSDBA/SYSDBA MODE='IN' CONTROL=\'/opt/data/test.ctl\'
```

例 2 使用 USERID 和 CONTROL 参数启动 dmfldr，完成载出。

```
dmfldr USERID=SYSDBA/SYSDBA MODE='OUT' CONTROL=\'/opt/data/test.ctl\'
```

例 3 使用 USERID 、OUT、TABLE 和 DATA 参数启动 dmfldr。将 test 表中数据载出到 test.txt 文件中。

```
dmfldr USERID=SYSDBA/SYSDBA@localhost:5236 MODE='OUT'  TABLE=test
DATA=\'/opt/data/test.txt\'
```

例 4 使用 USERID、IN、TABLE 和 DATA 参数启动 dmfldr。将 test.txt 文件中数据载入到 test 表中。

```
dmfldr USERID=SYSDBA/SYSDBA@localhost:5236 MODE='IN'  TABLE=test
DATA=\'/opt/data/test.txt\'
```

除了在启动命令行中直接指定 dmfldr 参数值外，用户还可以通过 CONTROL 参数设置控制文件中的 OPTIONS 选项来指定 dmfldr 参数值，也可以在 dmfldr.ini 配置文件中指定 dmfldr 参数值。

注意： **参数值的优先选择顺序为先控制文件中的OPTIONS选项，其次命令行中指定的参数值，最后dmfldr.ini配置文件中指定的参数值**

2.2 查看 dmfldr 参数

dmfldr 使用较为灵活，参数较多，用户可以使用“dmfldr help”查看 dmfldr 版本信息和各参数的简单信息。

```
dmfldr help

version: 1-2-101-21.12.16-153499-10000-ENT

格式: DMFLDR   KEYWORD=value

例程: DMFLDR   SYSDBA/SYSDBA CONTROL='c:\fldr.ctl'

USERID 必须是命令行中的第一个参数

字符串类型参数必须以引号封闭


关键字                  说明（默认值）

------------------------------------------------------------------------------

USERID                  数据库的连接信息

CONTROL                 控制文件，字符串类型

LOG                     日志文件，字符串类型 (fldr.log)

BADFILE                 错误数据记录文件，字符串类型 (fldr.bad)

SKIP                    初始忽略逻辑行数 (0)

LOAD                    需要装载的行数 (ALL)

ROWS                    提交频次 (50000), DIRECT 为 FALSE 有效

DIRECT                  是否使用快速方式装载 (TRUE)

SET_IDENTITY            是否插入自增列 (FALSE)

SORTED                  数据是否已按照聚集索引排序 (FALSE)
```

INDEX_OPTION	索引选项 (1) 1 不刷新二级索引，数据按照索引先排序，装载完后再 将排序的数据插入索引 2 不刷新二级索引，数据装载完成后重建所有二级索引 3 刷新二级索引，数据装载的同时将数据插入二级索引
ERRORS	允许的最大数据错误数 (100)
CHARACTER_CODE	字符编码，字符串类型 (GBK, UTF-8, SINGLE_BYTE, EUC-KR)
MODE	装载方式，字符串类型 IN 表示载入，OUT 表示载出， OUTORA 表示载出 ORACLE (IN)
CLIENT_LOB	大字段目录是否在本地 (FALSE)
LOB_DIRECTORY	大字段数据文件存放目录
LOB_FILE_NAME	大字段数据文件名称，仅导出有效 (dmfldr.lob)
BUFFER_NODE_SIZE	读入文件缓冲区的大小 (10),有效值范围 1~2048
LOG_SIZE	日志信息缓冲区的大小 (1),有效值范围 1~100
READ_ROWS	工作线程一次最大处理的行数 (100000)，最大支持 2^26-10000
NULL_MODE	载入时 NULL 字符串是否处理为 NULL 载出时空值是否处理为 NULL 字符串 (FALSE)
NULL_STR	载入时视为 NULL 值处理的字符串
SEND_NODE_NUMBER	运行时发送节点的个数 (20)，有效值范围 16~65535
TASK_THREAD_NUMBER	处理用户数据的线程数目，默认与处理器核数量相同，有效值范围 1~128
BLDR_NUM	服务器 BLDR 数目 (64),有效值范围 1~1024
BDTA_SIZE	bdta 的大小 (5000)，有效值范围 100~10000
COMPRESS_FLAG	是否压缩 bdta (FALSE)
MPP_CLIENT	MPP 环境，是否本地分发 (TRUE)
SINGLE_FILE	MPP/DPC 环境，是否只生成单个数据文件(FALSE)
LAN_MODE	MPP/DPC 环境，是否以内网模式装载数据(FALSE)
UNREP_CHAR_MODE	非法字符处理选项(0),为 0 时表示跳过该数据行，为 1 时表示使用(*)替换错误字节
SILENT	是否静默方式装载数据(FALSE)
BLOB_TYPE	BLOB 类型字段数据值的实际类型，字符串类型 (HEX_CHAR)

	HEX 表示值为十六进制，HEX_CHAR 表示值为十六进制字符类型 仅在 direct=FALSE 有效
OCI_DIRECTORY	OCI 动态库所在的目录
DATA	指定数据文件路径
ENABLE_CLASS_TYPE	允许用户导入 CLASS 类型数据 (FALSE)
FLUSH_FLAG	提交时是否立即刷盘 (FALSE)
IGNORE_BATCH_ERRORS	是否忽略错误数据继续导入 (FALSE)
SINGLE_HLDR_HP	是否使用单个 HLDR 装载 HUGE 水平分区表 (TRUE)
EP	指定需要发送数据的站点序号列表，仅向 MPP/DPC 环境导入数据时有效
PARALLEL	是否开启并行装载(FALSE)
SQL	使用自定义查询语句，仅导出模式有效
SQLFILE	自定义查询语句所在文件，仅导出模式有效
TABLE	导入/出表
ROW_SEPERATOR	行分隔符
FIELD_SEPERATOR	列分隔符
COMMIT_OPTION	提交选项(0)，0:每发送一批数据后提交，1:发送完所有数据后提交
APPEND_OPTION	追加选项(0)，0：追加方式，1：替代方式，2：插入方式
COLNAME_HEADING	是否在导出文件头中打印列名(FALSE)
IGNORE_AIMLESS_DATA	是否忽略无目标数据(FALSE)
LOB_AS_VARCHAR	是否将 CLOB 作为 VARCHAR 进行导入导出(FALSE)
LOB_AS_VARCHAR_SIZE	将 CLOB 作为 VARCHAR 进行导入导出时，lob 数据最大大小(10)MB
LOG_LEVEL	记录错误数据信息级别(3)，0：不记录 1：只记录到 log 文件 2：只记录到 bad 文件 3：记录到 log 和 bad 文件 4：错误仅输出到屏幕
FLDR_INI	配置文件路径，字符串类型
RECONN	自动重连次数(0)
RECONN_TIME	自动重连等待时间(5)，单位(s)，有效值范围(1~10000)
WIDTH	设置列数据宽度
SEDF	被替换的字符列表
SEDT	用于替换的字符列表
ESCAPE	转义符

HELP	打印帮助信息

2.3dmfldr 参数简介

dmfldr 的众多参数中，USERID 是必选参数，且位置必须是第一位。其余参数均为可选参数，需要时指定，指定时也无顺序要求。

- **USERID**

USERID 用于指定数据库的连接信息。必选参数，且必须位于参数位置的第一个。

语法如下：

```
{{<username>[/<password>]} | /}[@<connect_identifier>][<option>] [<os_auth>]
<connect_identifier> ::=<svc_name> | {<host>[:<port>]} | <unixsocket_file>
<option>::= #{ <exetend_option>=<value>{,<extend_option>=<value>} }  //此行外层{}是为了封装参数之用，书写时需要保留
<os_auth>::= AS {SYSDBA|SYSSSO|SYSAUDITOR|USERS|AUTO}
```

{{<username>[/<password>]} | /}：<username>[/<password>]为用户名和密码。普通登录方式时用户名必写，密码缺省为 SYSDBA。/表示采用操作系统身份验证方式登录或利用 wallet 文件登录。采用操作系统身份验证方式登录时无需指定用户名和密码，即使指定也会被忽略。利用 wallet 文件登录时，dm_svc.conf 文件中的配置项 WALLET_LOCATION 必须非空，客户端会通过用户输入的服务名以及 WALLET_LOCATION 配置项指定的 wallet 文件路径自动获取 wallet 文件中服务名所对应的用户名和密码，因此用户无需输入用户名和密码，若用户输入了用户名和密码，则优先使用用户输入的用户名和密码登录数据库，关于利用 wallet 文件登录数据库的更多详细介绍请参考手册《DM8 安全管理》。

<svc_name>：服务名。服务名在 dm_svc.conf 中配置。dm_svc.conf 的配置请参考《DM8 系统管理员手册》。

<host>[:<port>]：服务器 IP 地址和端口号。缺省情况下默认为本地服务器和端口号 LOCALHOST:5236。当服务器为本机时，SERVER:PORT 可直接写 LOCALHOST。当连接其他服务器时，SERVER:PORT 需写上 IP 地址和 PORTNUM，例如：192.168.0.248:8888。

<unixsocket_file>：专门用于在 LINUX 系统中，当服务器与客户端之间使用 UNIXSOCKETUNIX-IPC 方式通信时，指定客户端连接的 UNIXSOCKET 路径文件名。必须和 inet_type=UNIXSOCKET 同时使用。

例 使用 USERID 启动 dmfldr。

```
./dmfldr SYSDBA/SYSDBA@/home/test/foo.sock#{inet_type=UNIXSOCKET}
control=\'/home/test/dmfldr_test.ctrl\'
```

<option>为扩展选项，用法为<exetend_option>=<value>。所有 value 值不能包含空格，不能包含特殊的符号，如引号等。书写扩展选项时需要用引号#"{ }"进行封装，例如：#"{INET_TYPE=tcp,mpp_type=local}"。

表 2.1 exetend_option 扩展选项

extend_option	value
mpp_type	MPP 登录属性，此属性的设置对非 MPP 系统没有影响。取值 GLOBAL 和 LOCAL，默认为 GLOBAL。GLOBAL 表示 MPP 环境下建立的会话为全局会话，对数据库的导入导出操作在所有节点进行；LOCAL 表示 MPP 环境下建立的会话为本地会话，对数据库的导入导出操作只在本地节点进行
inet_type	网络通信协议类型。取值 UDP/TCP/IPC/RDMA/UNIXSOCKET。缺省为 TCP
ssl_path	通信加密的 SSL 数字证书路径，缺省为不使用加密。数字证书路径由用户自己创建，将相应的证书需放入该文件夹中。其中服务器证书必须与 dmserver 目录同级，客户端目录可以任意设置
ssl_pwd	通信加密的 SSL 数字证书密码。和 ssl_path 一起使用
server_option	服务器扩展串，用于后续扩展只有服务器需要的参数选项设置。书写服务器扩展串需要用{}进行封装。例如： server_option={ <opt>=<val>{,<opt>=<val>} }。

例 一个包含扩展选项的、完整的例子。

```
./dmfldr
SYSDBA/SYSDBA@192.168.1.64:5236#"{mpp_type=local,inet_type=tcp,server_option=
{opt1=123,opt2=456}}" control=\'/home/test/dmfldr_test.ctrl\'
log=\'/home/test/log/fldr.log\'
```

SSLPATH@SSLPWD：通信加密的 SSL 数字证书路径和密码，缺省为不使用加密。数字证书路径由用户自己创建，将相应的证书需放入该文件夹中。其中服务器证书必须与 dmserver 目录同级，客户端目录可以任意设置。

AS <SYSDBA|SYSSSO|SYSAUDITOR|USERS|AUTO>：操作系统身份验证。用户可以通过将操作系统用户加入到操作系统的 dmdba|dmsso|dmauditor 用户组来使用操作系统用户登录数据库，分别对应数据库的 SYSDBA|SYSSSO|SYSAUDITOR 用户。还可以通过将操作系统用户加入到操作系统的 dmusers 用户组来使用操作系统用户登录数据库，对应数据库的同名用户。AUTO 表示按顺序自动匹配数据库用户类型。操作系统身份验证无需输入用户名和密码，若输入用户名和密码将会被忽略。操作系统身份验证仅在 DM 安全版本中才提供支持，详情请参考《DM8 安全管理》。

- **CONTROL**

控制文件的路径，字符串类型。

控制文件用于指定数据文件的路径和数据格式。

在数据载入时，dmfldr根据控制文件指定的格式来解析数据文件；导出数据时，dmfldr也会根据控制文件指定的列分隔符、行分隔符等生成数据文件。控制文件中还可以指定其他的一些dmfldr参数值。缺省列分隔符是|，行分隔符是回车。对控制文件格式的详细介绍见下一章。

此参数为可选参数。

- **LOG**

dmfldr 的日志文件路径，字符串类型。默认日志文件名为 fldr.log。日志文件记录了 dmfldr 运行过程中的工作信息、错误信息以及统计信息。

此参数为可选参数。

- **NULL_STR**

指定数据文件中 NULL 值的表示字符串，字符串类型，默认忽略此参数。

若设置了 NULL_STR，则此参数值将成为数据文件中 NULL 值的唯一表示方式。NULL_STR 区分字符串大小写，并且长度不允许超过 128 个字节。

此参数为可选参数。

- **BADFILE**

记录错误数据的文件路径，字符串类型。默认的错误文件名为 fldr.bad。文件记录的信息为数据文件中存在格式错误的行数据以及转换出错的行数据。

用户也可以通过设置控制文件中的 OPTIONS 选项来指定错误数据文件的路径，同时也可以在控制文件的 LOAD 节点中指定错误数据文件的路径。BADFILE 选项的最终值的优先选择顺序为控制文件的 LOAD 节点选项，控制文件的 OPTIONS 选项，命令行中指定的参数

值，dmfldr.ini 配置文件中指定的参数值，用户可以同时对四种设置方式中的一个或多个设置，但最终的值只取一个。

此参数为可选参数，作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

- **SKIP**

跳过数据文件起始的逻辑行数，整型数值。默认的跳过起始行数为 0 行。如果用户指定了多个文件，且起始文件中的行数不足 SKIP 所指定的行数，则 dmfldr 工具会扫描下一个文件直至累加的行数等于 SKIP 所设置的行数或者所有文件都已扫描结束。

在 MPP 和分布计算集群 DPC 环境下，由于数据分散在各个节点，每个节点返回数据的速度也不一样，无法确定 SKIP 的数据，因此 SKIP 参数在 MPP 和分布计算集群 DPC 环境下无效。若在 MPP 环境下使用本地连接，则 SKIP 参数仍有效。

- **LOAD**

装载的最大行数，整型数值。默认的最大装载行数为数据文件中的所有行数。LOAD 指定的值不包括 SKIP 指定的跳过的行数。

在 MPP 和分布计算集群 DPC 环境下，由于数据分散在各个节点，每个节点返回数据的速度也不一样，无法确定 LOAD 的数据，因此 LOAD 参数在 MPP 和分布计算集群 DPC 环境下无效。若在 MPP 环境下使用本地连接，则 LOAD 参数仍有效。

- **ROWS**

每次提交的行数，整型数值。默认的提交行数为 50000 行。提交行数的值表示提交到服务器的行数，并不一定代表按照数据文件中的数据顺序的行数。用户可以根据实际情况调整每次提交的行数，以达到性能的最佳点。

此参数为可选参数，作用于 MODE 为 IN 且 DIRECT 为 FALSE 的情况下，其他情况下无效。

- **DIRECT**

数据装载方式，布尔值。缺省为 TRUE。DIRECT 指定了装载的方式，当为 TRUE 时，dmfldr 选择快速的载入模式，通过数据的转换和数据的封装直接对 B 树进行操作，省去了普通插入方式下各个操作符之间的跳转，提升了装载的效率，但对于约束的检查等由用户保证，dmfldr 将不处理有约束冲突的数据。当为 FALSE 时，dmfldr 选择普通的插入方式装载数据，可以保证数据的正确性和约束的有效性，效率比前者要低。

此参数为可选参数，作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

- **SET_IDENTITY**

设置自增列选项，布尔值。缺省为 FALSE。如果指定 SET_IDENTITY 选项值为 TRUE，则 dmfldr 将把从数据文件中读取的自增列值作为目标值插入数据库表中，用户应当保证每一行的自增列的值符合自增列的规则。如果 SET_IDENTITY 选项值设置为 FALSE，则 dmfldr 将忽略数据文件中对应自增列的值，服务器将自动生成自增列的值插入每一行的对应列。

此参数为可选参数，仅在 MODE 为 IN、DIRECT 为 TRUE 且非 DMDPC 环境下有效，DMDPC 环境下暂不支持自增列装载。

- **SORTED**

数据是否已经按照聚集索引排序，布尔值。缺省为 FALSE。如果设置为 TRUE，则用户必须保证数据已按照聚集索引排序完成，并且如果表中存在数据，需要插入的数据索引值要比表中数据的索引值大，服务器在做插入操作时顺序进行插入。如果设置为 FALSE，则服务器对于每条记录进行定位插入。

此参数为可选参数，作用于 MODE 为 IN 且 DIRECT 为 TRUE 的情况下，对于其他情况此参数无效。

- **INDEX_OPTION**

索引的设置选项，整型数值。缺省为 1。INDEX_OPTION 的可选项有 1、2 和 3。1 代表服务器装载数据时先不刷新二级索引，而是将新数据按照索引预先排序，在装载完成后，再将排好序的数据插入索引；2 代表服务器在快速装载过程中不刷新二级索引数据，只在装载完成时重建所有二级索引；3 代表服务器使用追加模式来进行二级索引的插入，在数据装载的过程中，同时进行二级索引的插入。若未禁用全局索引，则全局索引可通过设置 INDEX_OPTION 为 3 进行插入。

此参数为可选参数，作用于 MODE 为 IN 且 DIRECT 为 TRUE 的情况下，对于其他情况此参数无效。

- **ERRORS**

最大的容错个数，整型数值。取值范围为 0~4294967295，缺省为 100。当 dmfldr 在装载过程中出现错误的个数超过了 ERRORS 所设置的数目，则 dmfldr 将当前时间点已成功装载且未提交的所有正确数据提交到服务器，随后结束装载。如果载入过程中不允许出现错误则可以设置 ERRORS 为 0，如果允许所有的错误出现，则可以设置 ERRORS 为一个非常大的数。ERRORS 所统计的错误包含在数据转换和数据插入过程中所产生的数据错误。

此参数为可选参数，作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

■ **CHARACTER_CODE**

数据文件中数据的编码格式，字符串类型。默认与当前机器编码格式一致。CHARACTER_CODE 的可选项有 GBK、GB18080、UTF-8、SINGLE_BYTE 和 EUC-KR 五种：GBK 和 GB18030 对应中文编码；UTF-8 对应 UTF-8 国际编码；SINGLE_BYTE 对应单字节字符编码；EUC-KR 对应韩文字符集。

在 MODE 为 IN 的情况下，指定编码格式为 SINGLE_BYTE 时，dmfldr 将不做字符完整性检查，按单字节顺序读取每个字符。指定编码格式为 GBK、GB18030 或 UTF-8 时，dmfldr 将对每一个字符做字符的完整性检查，确保数据的正确性。

用户在使用 dmfldr 时可以根据不同的数据文件调整编码的格式确保装载的正确性，同时如果可以确保数据文件中的数据为单字节字符，则指定 SINGLE_BYTE 的载入效率将优于指定其他字符集的情况。

当 MODE 为非 IN 时，不支持 SINGLE_BYTE。

■ **MODE**

dmfldr 的装载模式，字符串类型。缺省值为 IN。MODE 的可选项有 IN、OUT、OUTORA 和 LOCAL_FILE 四种。

IN 模式指从数据文件中将数据装载入数据库，这种模式下控制文件的格式对应为数据文件中现有数据的格式；OUT 模式指从数据库中将数据导出到数据文件，这种模式下控制文件所指定的格式为数据存放在数据文件中的格式。需要说明的是在 OUT 模式下，如果指定了多个数据文件，则 dmfldr 最终只会将数据写入第一个数据文件。

OUT 模式下的控制文件与 IN 模式下的控制文件格式相同，用户可以通过使用同一个文件进行载入和导出数据。

OUTORA 模式表示导出 ORACLE 表的数据，此模式下暂不支持带有大字段表的导出。

LOCAL_FLIE 表示数据导入时，不向服务器发送数据，而是将数据写入成本地文件。通常用于多站点场景下，将数据按站点分类重新组织数据。

■ **CLIENT_LOB**

指明 LOB_DIRECTORY 表示的目录是否是客户端本地目录，布尔类型，缺省值为 FALSE。

CLIENT_LOB 仅在 MODE 为 IN 且 DIRECT 为 TRUE 时有效，指明在载入大字段对象数据时，LOB_DIRECTORY 参数指定的目录是否是客户端本地目录。若为 TRUE，表示目录

为客户端本地目录，且此时待载入的大字段长度不得超过 2G，否则无法成功载入大字段；若为 FALSE，表示目录为 DM 服务器所在主库的目录。

- **LOB_DIRECTORY**

指明 dmfldr 使用的大字段数据存放的目录，字符串类型。

当 MODE 为 OUT 时，dmfldr 生成大字段对应的数据文件，文件名由 LOB_FILE_NAME 指定，并存放于 LOB_DIRECTORY 指定的目录，如果未指定 LOB_DIRECTORY 则存放于指定的导出数据文件同一目录。对于 MODE 为 OUT 的情况，此参数为可选参数。

当 MODE 为 IN 且 DIRECT 为 TRUE 时，此时数据载入若涉及到大字段对象，需要用户指定大字段数据文件。若 CLIENT_LOB 为 TRUE，LOB_DIRECTORY 应指定大字段数据文件所在的客户端本地目录；若 CLIENT_LOB 为 FALSE，用户必须先把相关文件传送到 DM 服务器所在主库，然后使用 LOB_DIRECTORY 指明存放目录。此时此参数为必选参数。

当 MODE 为 IN 且 DIRECT 为 FALSE 时，此参数无效。

- **LOB_FILE_NAME**

指明 dmfldr 导出大字段数据的文件名，字符串类型，缺省为 dmfldr.lob。

当 MODE 为 OUT 时，dmfldr 生成大字段对应的数据文件名由 LOB_FILE_NAME 指定，若未指定默认为 dmfldr.lob，文件存放于 LOB_DIRECTORY 指定的目录。

此参数为可选参数，作用于 MODE 为 OUT 的情况下，当 MODE 为 IN 时无效。

- **BLOB_TYPE**

指定 BLOB 数据值的实际类型，字符串类型。可选项为 HEX 表示值为十六进制，HEX_CHAR 表示值为十六进制字符，缺省为 HEX_CHAR。

此参数为可选参数，只在 DIRECT 参数为 FALSE 的情况下有效。

注意： **不支持使用包含BLOB列的表生成的导出数据文件在导入时指定BLOB_TYPE为HEX_CHAR。**

- **BUFFER_NODE_SIZE**

指定读取文件缓冲区页大小，整数类型，单位为 MB，取值范围为 1~2048，缺省为 10。

BUFFER_NODE_SIZE 的值越大，缓冲区的页越大，每次读取的数据就越多，每次发送到服务器的数据也就越多，效率越高。但其大小受 dmfldr 客户端内存大小限制。

- **LOG_SIZE**

指定日志系统中缓冲区结点的大小，整数类型，单位为 MB，取值范围为 1~100，缺省为 1。

在装载过程中产生的错误数据和错误描述信息，将首先写入到日志系统的缓冲区结点中，缓冲区写满后，将缓冲区中的错误数据刷盘到错误数据文件中，并将缓冲区中的错误描述信息刷盘到日志文件中。若错误数据或错误描述信息的大小超过缓冲区结点的大小，则系统直接将该错误数据或错误描述信息刷盘到错误数据文件或日志文件中。

- **READ_ROWS**

指定读取缓冲区每次读取的最大行数，整数类型，取值范围为 0~(2^{26}-10000)，缺省为 100000。

在某些情况下，1MB 的 BUFFER_NODE_SIZE 读入的数据行数很大，而后续操作处理不了这么大的行数，此时可以用 READ_ROWS 来限制处理的行数。dmfldr 取 READ_ROWS 和 BUFFER_NODE_SIZE 中较小的值作为一次处理的行数。

- **NULL_MODE**

指定载入和导出数据时对 NULL 字符串和空值的处理方式，布尔类型，缺省为 FALSE。

若 NULL_MODE 为 TRUE，数据载入时将 NULL 字符串处理为空值，数据导出时则空值处理为 NULL 字符串；若设置为 FALSE，数据载入时将 NULL 字符串处理为字符串，数据导出时将空值处理为空串。

- **SEND_NODE_NUMBER**

指定 dmfldr 在数据载入时发送节点的个数，整数类型，取值范围为 16~65535，缺省有 20 个数据节点。

此参数为可选参数，只在 MODE 为 IN 的情况下有效。

- **TASK_THREAD_NUMBER**

指定 dmfldr 在数据载入时处理用户数据的线程数目，整数类型，取值范围为 1~128。默认情况下，dmfldr 将该参数值设为系统 CPU 的个数，但无论设定值是多少，dmfldr 至少会创建 2 个任务线程。在多核 CPU 环境下，增大 TASK_THREAD_NUMBER 值可以提升 dmfldr 装载性能。

说明： **在导出模式下，当 TASK_THREAD_NUMBER 设置为大于 16 而小于 128 时，dmfldr 不会报错，但会将 TASK_THREAD_NUMBER 自动置为 16。**

- **BLDR_NUM**

水平分区表装载时，指定服务器 BLDR 的最大个数，整数类型，取值范围为 1～1024，缺省为 64。

服务器的 BLDR 保存水平分区子表相关信息，BLDR_NUM 的设置也就指定了服务器能同时载入的水平分区子表的个数。

此参数为可选参数，作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

- **BDTA_SIZE**

BDTA（Batch DaTA）的大小，整数类型，取值范围为 100~10000，缺省为 5000。

BDTA 代表 DM 数据库批量数据处理机制中一个批量，在内存、CPU 允许的条件下，增大 BDTA_SIZE 能加快装载速度；在网络是装载性能瓶颈时，增大 BDTA_SIZE 影响不大。

此参数为可选参数，作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

- **COMPRESS_FLAG**

指定是否压缩 BDTA，布尔类型，缺省为 FALSE。

压缩 BDTA 能节省网络带宽，但同时也会加重 CPU 的负担，用户应根据具体应用情况考虑是否制定压缩。

- **MPP_CLIENT**

指定当服务器环境为 MPP 环境时 dmfldr 的数据装载模式，布尔类型，缺省为 TRUE。

当服务器环境为 MPP 环境时，若 MPP_CLIENT 为 TRUE，为客户端分发模式，数据在 dmfldr 客户端分发好后直接往指定站点发送数据；若 MPP_CLIENT 为 FALSE，为本地分发模式，dmfldr 客户端将数据全部发往连接的 MPP EP 站点。

MPP 环境下要配置 dmmal.ini 文件中的 MAL_INST_HOST 和 MAL_INST_PORT 参数。

此参数只有当服务器环境为 MPP 环境时才有效。

- **SINGLE_FILE**

指定当服务器环境为 MPP/DPC 环境时，dmfldr 的导出数据文件是否为单个文件，布尔类型，默认为 FALSE。

此参数只在 MPP/DPC 环境下且 MODE 为 OUT 时有效。参数值为 TRUE 表示仅生成一个数据导出文件，MPP/DPC 各个站点的数据将导出到同一个数据文件；值为 FALSE 表示可以生成多个数据文件，每一个 MPP/DPC 站点都有专门的数据文件接收该站点的数据。

- **LAN_MODE**

指定当服务器环境为 MPP/DPC 环境时，dmfldr 导入/导出数据是否使用局域网，布尔类型，缺省为 FALSE。

此参数只在 MPP/DPC 环境下有效。值为 TRUE 表示使用局域网，此时服务器的 MAL 系统必须配置了局域网 IP，否则 dmfldr 依然采用服务器对外服务的外网 IP；值为 FALSE 表示不使用局域网。

- **UNREP_CHAR_MODE**

指定是否将不完整的字符用“*”替换。1 是，0 否，缺省为 0。

设置为 1 时，dmfldr 装载过程中遇到包含不完整字符的数据时，将用"*"替换不完整的字符；为 0 时，该行数据可能会被丢弃。

- **SILENT**

指定是否以静默方式进行数据装载，布尔类型，缺省为 FALSE。

当设置为 TRUE 时，dmfldr 装载过程中将忽略反馈式的消息，例如装载进度提示信息、错误信息等，但仍然会在装载的开始和结束阶段打印一些静态/统计信息。静默方式只作于客户端的屏幕打印，dmfldr 日志依然会完整地记录装载过程中的详细信息。

- **OCI_DIRECTORY**

指定 ORACLE OCI 动态库所在的目录，字符串类型。

该参数与 MODE 参数配合使用，当 MODE 指定为 OUTORA 时，使用 OCI_DIRECTORY 指定 OCI 的动态库来构建导出环境。

- **DATA**

指定数据文件路径，字符串类型。

一般情况下数据文件路径在控制文件中指定。如果控制文件缺省或控制文件中数据文件路径指定为‘*’，那么会使用命令行或 dmfldr.ini 配置文件中 DATA 参数指定的数据文件。优先使用命令行中指定的 DATA 参数指定的数据文件。

- **ENABLE_CLASS_TYPE**

指定是否以 BLOB 方式载入或导出 CLASS 类型列数据，布尔类型，默认为 FALSE。

CLASS 类型是 DM 数据库服务器内部实现的数据类型，实际以 BLOB 类型存储。如果通过交互式工具或 DM 提供的接口等正常途径创建的 CLASS 类型数据，内部会转换成 BLOB 存储。而通过 dmfldr 直接导 CLASS 数据，没有进行转换，有可能出现载入的大对象数据无法转换成对应的 CLASS 类型。因此，当将 ENAME_CLASS_TYPE 设为 TRUE 时，用户要保证对应的 BLOB 数据能正确转换成对应的 CLASS 类型。

- **FLUSH_FLAG**

指定提交数据时服务器的处理方式，布尔类型，缺省为 FALSE。

该参数用于 dmfldr 向服务器发送 commit 请求时，是否要求服务器提供可靠的服务响应，若设置为 TRUE，则服务器会等数据写入磁盘后才将响应结果返回给 dmfldr，此种方式会降低数据装载的效率，但提供可靠的服务，不存在数据丢失的情况；若为 FALSE，则服务器在确认数据正确无误后便将响应结果返回，随后再写入磁盘，此种方式装载效率高，但若遇到掉电、机器故障、服务器崩溃等灾难性情况下，有可能会导致数据来不及写入磁盘而导致数据丢失。

此参数为可选参数，作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

- **SINGLE_HLDR_HP**

是否使用单个 HLDR 装载 HUGE 水平分区表主表。取值 TRUE 或 FALSE，缺省值为 FALSE。TRUE 表示使用单个 HLDR 装载 HUGE 水平分区表主表。FALSE 表示装载涉及到的每个叶子子表都各使用一个 HLDR。

HLDR 是服务器端装载 HUGE 表时用到的处理装置。SINGLE_HLDR_HP 的设置相当于指定了服务器装载 HUGE 水平分区主表时可以使用资源的模式。

TRUE 的方式更节约空间，FALSE 的方式装载速度更快，用户需要根据自己得需要权衡哪种方式更适合自己。

- **IGNORE_BATCH_ERRORS**

指定用户在数据装载过程中遇到数据错误时，是否忽略错误继续装载，还是报错返回，布尔类型，缺省为 FALSE。

该参数只在 DIRECT 为 FALSE 时有效。在普通数据装载方式下，以绑定插入的方式向服务器发送数据，服务器检查 IGNORE_BATCH_ERRORS 参数，发现为 TRUE 时，将处理所有接受到的数据，如果有不合要求的数据，服务器保存该数据的错误信息，直到当前批次所有数据处理完毕后，再将执行的结果返回 dmfldr，dmfldr 根据服务器返回的错误信息向错误数据文件写入错误数据。当 IGNORE_BATCH_ERRORS 参数值为 FALSE 时，服务器一旦遇到不合要求的数据，则终止当前批次数据的装载，向 dmfldr 返回错误信息，dmfldr 则直接废弃当前批次的所有数据。

此参数为可选参数，作用于 MODE 为 IN 且 DIRECT 为 FALSE 的情况下，当 MODE 为 OUT 时无效。

- **EP**

指定 dmfldr 只能向指定的 MPP/DPC 站点发送数据，对于其他站点的数据直接丢弃。EP 站点序号用整型数字表示，各站点间用逗号分隔，整个 EP 参数值用括号括起来。这是 EP 参数的一个常用用法：EP=(1,3,4)，表示向站点号为 1、3、4 的站点发送数据。

该参数仅在 MPP/DPC 环境有效，dmfldr 只能向 EP 指定的站点上的表上锁并发送数据，其他站点的表操作与其无关。需要注意的是，指定 EP 方式对于随机分布表无效，并且 dmfldr 当前连接站点的数据发送不能由其他 dmfldr 客户端发送，换言之，客户端 A 连接 EP1，客户端 B 连接 EP2，那么 EP1 的表数据只能有 A 来发送，不能由 B 发送，否则会导致锁等待或锁超时错误。

此参数为可选参数，在 MODE 为 IN 且 DIRECT 为 TRUE 的情况下有效。

- **PARALLEL**

dmfldr 向服务器导入数据时，是否开启并行模式。

开启并行参数后，可以运行多个用户同时向同一张表装载数据，此模式可以充分利用客户端的数据处理能力，服务器接收到各个客户端发送过来的数据后，再进行统一的数据处理和刷盘。若存在以下情况，并行参数不可开启：

1． 装载表存在二级索引；

2． 装载表没有自定义的聚集索引；

3． 装载表开启了逻辑日志或者高级日志；

4． 装载表为列存储表(HUGE 表)。

此参数为可选参数，在 MODE 为 IN 且 DIRECT 为 TRUE 的情况下有效。

- **SQL**

用户指定 SQL 查询语句，用于导出自定义查询结果数据，仅导出模式有效。

自定义 SQL 查询语句会将 SQL 语句发送到服务器进行分析执行，若发现 SQL 语句不是查询语句，dmfldr 将报错返回。SQL 语句可以实现用户的个性化数据导出需要，可以选择需要导出的列，需要过滤的结果。

此参数为可选参数，在 MODE 为 OUT 的情况下有效。

SQL 查询语句需要使用双引号括起来。

例 在导出过程中使用 SQL 查询语句。

```
./dmfldr USERID=SYSDBA/SYSDBA@192.168.100.121:5254 MODE=\'OUT\' SQL="select *
from test where c1='b';" data=\'/home/test/yy/test.txt\'
```

- **SQLFILE**

用户指定 SQL 查询语句所在文件路径，用于导出自定义查询结果数据，仅导出模式有效。

当 SQL 查询语句过长时，用户可以将 SQL 查询语句写入文件中，然后使用 SQLFILE 参数指定 SQL 查询语句所在文件路径。

此参数为可选参数，在 MODE 为 OUT 的情况下有效。

例 SQL 查询语句所在文件路径为/home/test/yy/test.sql，文件内容如下：

```
select * from test where c1='b';
```

使用 SQLFILE 参数指定 SQL 查询语句所在文件路径进行数据导出。

```
./dmfldr USERID=SYSDBA/SYSDBA@192.168.100.121:5254 MODE=\'OUT\'
SQLFILE=\'/home/test/yy/test1.txt\' data=\'/home/test/yy/test.txt\'
```

- **TABLE**

指定导入或导出表的表名，可以包含模式名。可选参数。

若导出时指定 SQL 或 SQLFILE 参数，则该选项失效。

- **ROW_SEPERATOR**

指定行分隔符，分隔符额外指定 X 表示十六进制的分隔符，长度应小于 128。可选参数。

- **FIELD_SEPERATOR**

指定列分隔符，分隔符额外指定 X 表示十六进制的分隔符，长度应小于 255。可选参数。

例 使用 FIELD_SEPERATOR 参数指定分隔符。

```
./dmfldr USERID=SYSDBA/SYSDBA@192.168.100.165:9999  FIELD_SEPERATOR=\'|\'
ROW_SEPERATOR=X '0A' DATA= \'/opt/data/D1.TXT\' TABLE="sysdba".\"tyjm\"
mode='in' SINGLE_FILE=true
```

- **COMMIT_OPTION**

提交选项。 取值：0 表示每发送一批数据进行提交； 1 表示数据全部装载完毕后进行提交。缺省为 0。可选参数，导入时有效。

- **APPEND_OPTION**

dmfldr 的追加模式。取值：0 表示 APPEND 追加方式； 1 表示 REPLACE 替代方式；2 表示 INSERT 插入方式。缺省为 0。

当 dmfldr 处于数据装载模式时： APPEND 追加方式，在表中追加新记录；REPLACE 替代方式，先清空表再插入新记录；INSERT 插入方式，向空表插入新记录（如果不是空表则会报错无效的装载模式）。

当 dmfldr 处于导出数据模式时：dmfldr 会检查导出数据文件是否存在，若不存在则直接创建新文件；若存在，设置为 APPEND 追加方式时，以追加的方式写入数据；设置为 REPLACE 替代方式时，先删除文件再重新创建新文件；设置为 INSERT 插入方式则报错。

- **COLNAME_HEADING**

是否在导出文件头中打印列名。取值 TRUE 或 FALSE，缺省值为 FALSE。TRUE 表示在导出文件头中打印列名，列名之间使用列分隔符进行分隔，列名与数据之间使用行分隔符进行分隔。可选参数，导出时有效。

- **IGNORE_AIMLESS_DATA**

是否忽略无目标数据。取值 TRUE 或 FALSE，缺省值为 FALSE。TRUE 表示在导入数据时忽略无目标数据，即未找到对应分区的分区表数据，该错误不影响其他数据的导入，相关错误个数不会计入 ERRORS，并且不会显示相关错误信息，导入结束后用户可通过提交总数、读取逻辑记录总数、拒绝逻辑记录总数以及跳过逻辑记录总数的差值来获取忽略的无目标数据的总数。可选参数，导入时有效。

- **LOB_AS_VARCHAR**

是否将 CLOB 作为 VARCHAR 进行导入导出。取值 TRUE 或 FALSE，缺省值为 FALSE。TRUE 表示进行数据导入导出时将 CLOB 类型数据作为 VARCHAR 类型数据处理，此时 CLOB 类型数据内容为实际数据内容。可选参数。

- **LOB_AS_VARCHAR_SIZE**

指定将 CLOB 作为 VARCHAR 进行导入导出时，支持的 LOB 数据的最大大小，整数类型，单位为 MB，取值范围为 1~20，缺省为 10。

可选参数，仅在 LOB_AS_VARCHAR 为 TRUE 时有效。

- **LOG_LEVEL**

指定记录错误数据信息的级别，可取值 0、1、2、3、4，缺省为 3。

0：不记录错误信息和错误数据；

1：仅记录错误信息；

2：仅记录错误数据；

3：记录错误信息和错误数据；

4：仅将错误输出到屏幕，不记录到文件中。

其中，错误信息指错误原因等信息，记录在日志文件（.lob 文件）中；错误数据指因为格式或数据本身错误而导致导入导出失败的数据，记录在错误文件（.bad 文件）中。

可选参数。

- **FLDR_INI**

指定 dmfldr.ini 配置文件路径，字符串类型。用户可在 dmfldr.ini 文件中指定除 USERID、CONTROL、HELP 以外的所有 dmfldr 参数值。

可选参数，未指定该参数时，默认在当前目录下查找 dmfldr.ini 配置文件。

dmfldr.ini 配置文件的语法如下所示：

```
OPTIONS(
<id>=<value>
……
)
```

文件中每个参数值对之间使用空格或者换行分隔。

dmfldr.ini 配置文件中指定 SQL 参数时，对于 SQL 语句中出现的单引号需要使用单引号括起来。

例 展示一个完整的 dmfldr.ini 文件。

```
OPTIONS(
MODE='OUT'
SQL='SELECT * FROM TEST WHERE C1=''B'';'
LOG='/home/test/yy/test_log.log'
DATA='/home/test/yy/test.txt'
)
```

- **RECONN**

指定自动重连次数，范围为 0~4294967295。缺省为 0，表示不进行自动重连。

在数据导入过程中，如果出现可恢复的环境错误（如主备切换、故障重启等）或网络错误（如网络超时、连接断开重连等）等问题，dmfldr 根据 RECONN 参数进行指定次数的自动重连处理，若自动重连成功，则 dmfldr 继续执行数据导入任务，若重连失败，则

dmfldr 根据 COMMIT_OPTION 采取相应的处理策略：若 COMMIT_OPTION=0，则未提交数据全部回滚；若 COMMIT_OPTION=1，则所有数据全部回滚。

此参数为可选参数，仅在 MODE 为 IN 的情况下有效。

- **RECONN_TIME**

指定自动重连等待时间，单位为 S，取值范围为 1~10000，缺省为 5。

当数据导入过程中出现可恢复的环境错误或网络错误等问题时，若 RECONN 不为 0，则 dmfldr 每间隔 RECONN_TIME 时间进行下一次自动重连。

此参数为可选参数，仅在 MODE 为 IN、RECONN 不为 0 的情况下有效。

- **WIDTH**

指定导出列的数据宽度，单位为字节，取值范围为 1~65535。不足指定宽度时填充空格，超出指定宽度时进行截断。未指定该参数时，导出列按实际长度导出。

不同列的指定宽度之间使用“:”进行分隔。若 WIDTH 指定的列个数少于实际导出列个数，则最后几列视为未设置 WIDTH，其导出列按实际数据长度导出。

此参数为可选参数，仅在 MODE 为 OUT 的情况下有效。

例 指定导出数据第一列的宽度为 12 字节，第二列的宽度为 30 字节。

```
./dmfldr USERID=SYSDBA/SYSDBA@192.168.100.121:5254 MODE=\'OUT\' SQL="select *
from test;" DATA=\'/home/test/yy/test.txt\' WIDTH=12:30
```

- **SEDF**

指定被替换的字符列表，以单个字符为单位进行替换，“0x”开头表示十六进制格式，系统自动将其转换为 ASCII 字符。

此参数为可选参数，与参数 SEDT 一同使用。

- **SEDT**

指定用于替换的字符列表，以单个字符为单位进行替换，“0x”开头表示十六进制格式，系统自动将其转换为 ASCII 字符。

将参数 SEDF 指定的字符列表逐个替换为 SEDT 指定的字符列表，参数 SEDF 和 SEDT 指定的字符个数必须相同。

此参数为可选参数，与参数 SEDF 一同使用。

例 导出数据时，将数据中的“b”和空格分别替换为“z”和“x”。

```
./dmfldr USERID=SYSDBA/SYSDBA@192.168.100.121:5254 MODE=\'OUT\' SQL="select *
from test;" DATA=\'/home/test/yy/test.txt\' SEDT=\'zx\' SEDF=\'b0x20\'
```

- **ESCAPE**

指定转义符，“0x”开头表示十六进制格式，系统自动将其转换为 ASCII 字符。仅支持指定一个转义符。

导出数据时，对于指定转义符会输出两个相邻的指定转义符；导入数据时，若存在两个相邻的指定转义符，则仅输入一个指定转义符。

此参数为可选参数。

例 指定转义符为“*”。

```
./dmfldr USERID=SYSDBA/SYSDBA@192.168.100.121:5254 MODE=\'OUT\' SQL="select *
from test;" DATA=\'/home/test/yy/test.txt\' ESCAPE=\'*\'
```

对于表中的数据“t*1”，将被转换为“t**1”。导入数据时，通过指定 ESCAPE 为“*”，将数据“t**1”转换为“t*1”。

- **HELP**

获取帮助信息。

3 dmfldr 实战

在上一章简单了解了 dmfldr 的各参数含义后，本章介绍如何通过设置不同参数值使用 dmfldr 进行不同情况的数据装载。并介绍了如何通过设置参数提高 dmfldr 的性能以及使用限制。

说明： **本章中皆以在 Linux 操作系统下进行操作为例，在 Windows 操作系统下目录分隔符与单引号转义的处理有些许区别，不再赘述**

3.1dmfldr 控制文件

控制文件 CONTROL 是用于指定数据文件路径和数据文件中数据的格式。在数据载入时，dmfldr 根据控制文件指定的格式来解析数据文件；导出数据时，dmfldr 也会根据控制文件指定的列分隔符、行分隔符等生成数据文件。

控制文件中还可以指定其他 dmfldr 参数值。

dmfldr 控制文件的语法如下所示：

```
[OPTIONS(
<id>=<value>
……
       )]
LOAD [DATA]
INFILE  < <file_option>|<directory_option> >
[BADFILE <path_name>]
[APPEND|REPLACE|INSERT]
<into_table_clause>

<id> ::=参数
<value> ::=值
```

```
<file_option> ::= [LIST] <path_name> [<row_term_option>] [,<path_name>
[<row_term_option>]]
<directory_option> ::= DIRECTORY <path_name> [<row_term_option>]
<path_name> ::=文件地址
<row_term_option> ::=STR [X] <delimiter>
<into_table_clause> ::= <into_table_single>{<into_table_single>}
<into_table_single> ::=INTO TABLE [<schema>.]<tablename>
                          [EP <ep_option>]
                          [WHEN <field_conditions>]
                          [FIELDS [TERMINATED BY] [X] <delimiter>]
                          [<enclosed_option>]
                          [TRAILING NULLCOLS]
                          [<coldef_option>]
<schema> ::=模式名
<tablename> ::=表名
<ep_option> ::=(<ep_list>)
<ep_list> ::=整型数字列表，以逗号分隔
<field_conditions> ::= <field_condition>{ AND <field_condition>}
<field_condition> ::= [(] <cmp_exp><cmp_ops><cmp_data>[)]
<cmp_exp> ::= <colid> | (p1:p2)
<cmp_ops> ::= = | <> | !=
<cmp_data> ::= [X] '<字符串常量>' | BLANKS | WHITESPACE
<delimiter> ::= '<字符串常量>'
<coldef_option> ::= (<col_def>{ ,<col_def>})
<col_def>::= <col_id>
             [FILLER][<property_option>][<dtype_option>][<fmt_option>][<term_
             option>] [<enclosed_option>][<constant_option>][<fun_option>]
<col_id> ::= 列名
<property_option> ::= <position_option> | NULL
<position_option> ::= position(p1:p2) | position(p1) | position(*)
```

```
<dtype_option> ::= DATE | CHAR

<fmt_option> ::= FORMAT '<日期格式>'

<term_option> ::= TERMINATED [BY] <wx_option>

<wx_option> ::= WHITESPACE|[X] <delimiter>

<enclosed_option> ::= [OPTIONALLY] ENCLOSE [BY] [X] <delimiter>

<constant_option> ::= CONSTANT "<常量>"

<fun_option> ::= "<函数>"
```

对于上述控制文件语法，需要说明的是：

- dmfldr 在处理数据文件中换行符时 windows 默认为 0x0D0A(\r\n)，非 windows 默认为 0x0A(\n)，用户应该根据现有的数据文件中的换行符做相应的调整。对应选项为<row_term_option>，若指定的<value>值为十六进制的字符串值需要指明[X]选项，<value>值不再需要以 0x 开头。若没有指明[X]选项，则<value>值为指定的字符串；
- 关于列分隔符，用户应当指定 FIELDS 或者 coldef_option 中的至少一种。若两者均存在，则以 coldef_option 中的设置为准，若分隔符指明[X]选项，则表明此分隔符为十六进制格式的字符串；
- 关于 file_option，用来指定单个文件；
- 关于 directory_option，用来指定整个文件夹。指定此选项后，dmfldr 会自动扫描指定文件夹下的所有文件，这些文件数据将被导入到服务器；
- 关于 LIST 选项，INFILE 使用 LIST 选项时，表明实际的数据文件路径存储在 INFILE 指定的文件中，该文件可以存储多个实际的数据文件路径，使用逗号或者换行分割；
- 关于 APPEND|REPLACE|INSERT 选项，当 dmfldr 处于数据装载模式时，INSERT 表示插入方式，向空表插入新记录（如果不是空表则会报错无效的装载模式）；APPEND 表示追加方式，为缺省方式，在表中追加新记录；REPLACE 表示替代方式，先清空表再插入新记录。

  当 dmfldr 处于导出数据模式时，dmfldr 会检查导出数据文件是否存在，若不存在则直接创建新文件；若存在，当设置为 APPEND 时，以追加的方式写入数据；设置为 REPLACE 时，先删除文件再重新创建新文件；设置为其他值时则报错。

选项默认值为 APPEND；

- 关于 OPTIONS 选项，该选项支持命令行参数中除 userid，control，help 以外的所有参数的指定，每个参数值对使用空格或者换行分割。对于 option 中出现的参数，在 dmfldr 的指定执行参数中也出现的，dmfldr 会选择 option 中对应参数的值执行；
- 关于 TRAILING NULLCOLS，该选项表示若数据文件中未提供列数据，则该列的值设为 NULL；
- 关于 col_def，FILLER 表示跳过处理数据文件中指定列的值；
- 关于 property_option 选项
  - ✓ position(p1:p2)：从数据文件中每行数据的第 p1 个字节到第 p2 个字节为该列值，包含边界 p1,p2；
  - ✓ position(p1)：从数据文件中每行数据的第 p1 个字节开始，到下一个列分隔符之间的数据为该列值，包含边界 p1；
  - ✓ position(*)：根据数据文件中前一列数据的结束位置确定当前列的起始位置，根据 dtype_option 选项确定当前列的结束位置。若当前列为第一列，则起始位置为第 1 个字节，若未指定 dtype_option 选项，则默认为 CHAR(1)；
  - ✓ position 选项对大字段数据无效，若对大字段类型指定此选项会报错；
  - ✓ NULL：指定的值为 NULL，忽略数据文件中的值；
  - ✓ 某一列使用了 position，后续的列都要使用 position；
  - ✓ property_option 参数仅对导入有效。
- 关于 dtype_option 选项，用来指定列的数据类型，仅当指定 position_option 选项时有效，目前仅支持 DATE 和 CHAR 两种类型，缺省为 CHAR；
- 关于 fmt_option 选项，用来指定日期格式，仅当 dtype_option 选项指定为 DATE 时有效。更多关于<日期格式>的介绍请参考《DM8_SQL 语言使用手册》中函数 TO_DATE/TO_TIMESTAMP/TO_TIMESTAMP_TZ；
- 关于 term_option 选项，该选项用来指定数据文件中指定列的结束标志。列的结束标志可以是 WHITESPACE（空格）或者用户自定义的字符串或十六进制串。指定了 term_option 后，该列不需要用 FIELDS 分隔；

- 关于 enclosed_option 选项，此参数指定封闭符，为可选参数，默认不存在封闭符。若在 into_table_clause 和 coldef_option 中均设置了封闭符，则以 coldef_option 中的设置为准，若封闭符前指定[X]选项，则表明此封闭符为十六进制格式的字符串。若数据列设置了封闭符，则将跳过封闭符取出数据；
- 分隔符或封闭符字符串的长度均不能超过 255 个字节；
- 关于 constant_option 选项，指定 constant 关键字后，数据文件中不需要为该列准备数据，如果指定了，该列数据将作为下一字段数据装载而导致数据混乱。constant 选项对大字段类型无效；
- 关于 fun_option 选项，目前只支持 trim()、replace(colname, srcStr, destStr)、sysdate 和 sequence.NEXTVAL。trim()函数用于去除列数据的前后空格；replace()函数用于将 colname 列名指定的列数据中的 srcStr 替换为 destStr，其中 srcStr 和 destStr 参数可使用 chr(int)函数将数字转换成字符串，要求 chr()的参数和返回值均不超过 4 个字节；sysdate 函数用于将系统当前时间插入指定列，忽略数据文件中指定列的值；sequence.NEXTVAL 用于将序列的 NEXTVAL 作为列数据，其中 sequence 为实际序列名，用户可在 sequence 前增加模式名，例如"sch1.seq1.NEXTVAL"表示使用 sch1 模式下的 seq1 序列的 NEXTVAL 作为列数据；
- 关于 ep_option 选项，用于指定数据将要发送的目的站点，仅适用于 MPP 环境。

例 展示一个完整的 dmfldr 控制文件。

```
OPTIONS
(
       SKIP = 0
       ROWS = 50000
       DIRECT = TRUE
       INDEX_OPTION = 2
)
LOAD DATA
INFILE '/opt/data/test1.txt' STR X '0A'
BADFILE '/opt/data/test1.bad'
INTO TABLE test1
```

```
FIELDS '|'
(
F1,
F2 DATE FORMAT 'YYYY-MM-DD',
F3 NULL,
F4 TERMINATED BY WHITESPACE ENCLOSE BY '(',
F5 CONSTANT "test",
F6 "trim()"
)

LOAD DATA
INFILE '/opt/data/test2.txt' STR X '0A'
BADFILE '/opt/data/test2.bad'
INTO TABLE test2
FIELDS '|'
(
C1 TERMINATED BY ' ',
C2,
C3 "sysdate"
)
```

在这个例子中：

- OPTIONS 选项中定义了 SKIP、ROWS、DIRECT 和 INDEX_OPTION 参数
- 有两个 LOAD 选项，表明有一次装载同时处理两个表，在每个 LOAD 选项中：
  - 指定了数据文件和数据文件的换行符，还指定了 BADFILE 文件
  - 指定了操作的数据库基表、列分隔符以及列定义

3.2 指定数据文件

指定数据文件的方式有三种：一是在控制文件中指定；二是通过命令行参数 DATA 直接指定；三是使用 dmfldr.ini 指定。使用其中一种方式指定即可。

当 dmfldr 工作在 IN 模式时，从数据文件中读取数据并载入数据库；当工作在 OUT 模式时，从数据库中将指定数据导出到数据文件。

数据文件通常为文本文件，列与列之间由列分隔符隔开，行与行之间由行分隔符隔开。数据文件中的列分隔符和行分隔符由用户指定，并在控制文件中设置为与数据文件中的一致。

3.2.1 在控制文件中指定数据文件

可以在控制文件的 LOAD 节点中指定数据文件。

例 展示如何使用控制文件进行载入。

1. 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT,C2 INT,C3 DATE);
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1 1|2015-11-06

2 2|2015-11-05

3 3|2015-11_04
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1 TERMINATED BY ' ',

C2,

C3 DATE FORMAT 'yyyy-mm-dd'

)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
```

3.2.2 使用 DATA 参数指定数据文件

也可以使用 DATA 参数指定 dmfldr 的数据文件。用户可以在命令行中直接指定 DATA 参数，也可以在 dmfldr.ini 配置文件中指定 DATA 参数。如果控制文件中数据文件路径指定为'*'，那么会使用命令行或 dmfldr.ini 配置文件中指定的 DATA 参数来替换'*'，优先使用命令行中指定的 DATA 参数进行替换。

例 展示如何使用 DATA 参数指定数据文件进行载入。

1. 建表 TEST。

```
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 INT,C3 DATE);
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1 1|2015-11-06
2 2|2015-11-05
3 3|2015-11_04
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE *
INTO TABLE test
FIELDS '|'
(
C1 TERMINATED BY ' ',
C2,
C3 DATE FORMAT 'yyyy-mm-dd'
)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
data=\'/opt/data/test.txt \'
```

3.3 数据转换与错误数据文件

dmfldr 使用的数据文件都是文本格式的，其中的列值都是以字符串的方式保存在数据文件中。要想将这些数据载入数据库表中，需要将字符串转换成数据库表各列对应的数据类型。dmfldr 支持所有 DM 数据库支持的列定义类型，包括字符串、数值、时间日期、时间日期间隔、大字段类型等。

若数据文件的编码方式与 DM 数据库服务器的编码方式不一样，dmfldr 还需要进行字符编码的转换。dmfldr 支持 UTF8、GBK 和 GB18030 编码之间的相互转换。

数据类型和编码转换工作由 dmfldr 客户端进行，在这个过程中如果出现错误，dmfldr 会跳过该行继续后面的工作，并记录错误行到 BADFILE 指定的文件。常见的出错的情况有以下几种：

- 编码转换失败
- 目标列为字符串类型时，数据长度大于列定义长度
- 目标列为数值类型时，数据包含非法字符或者转换后超出该数值的范围
- 目标列为日期类型时，dmfldr 默认按 yyyy-mm-dd hh:mi:ss 的格式解析，如果数据不是这样的格式，需要指定对应列的时间日期 fmt 格式

dmfldr 错误数据的文件路径由 BADFILE 参数设置，默认的错误文件名为 fldr.bad。用户也可以通过设置控制文件中的 OPTIONS 选项来指定错误数据文件的路径，同时也可以在控制文件的 LOAD 节点中指定错误数据文件的路径。错误数据文件路径最终值的优先选择顺序为控制文件的 LOAD 节点选项，控制文件的 OPTIONS 选项，命令行中指定的参数值，dmfldr.ini 配置文件中指定的参数值，用户可以同时对四种设置方式中的一个或多个设置，但最终的值只取一个。BADFILE 仅作用于 dmfldr 的工作 MODE 为 IN 的情况下，MODE 为 OUT 时无效。

当数据类型和编码转换中存在错误数据，而错误数在允许的最大错误数范围内时，dmfldr 会将出错的数据记录到错误数据文件中。文件记录的信息为执行程序、时间、目标表、数据文件中存在格式错误的行数据以及转换出错的行数据。

允许的最大容错个数由 ERRORS 选项设置，取值范围为 0~4294967295 之间的整数，缺省为 100。当 dmfldr 客户端在数据类型和编码转换过程中出现的错误个数超过了 ERRORS 所设置的数目，dmfldr 会停止载入，当前时间点的所有正确数据将会被提交到服

务器端。如果载入过程中不允许出现错误则可以将 ERRORS 设置为 0；如果允许所有的错误出现，则可以将 ERRORS 设置为一个非常大的数。

例 展示错误数据文件的用法。

1. 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT,C2 INT,C3 DATE);
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1 1|2015-11-06

2 2|2015-11-05

3 3|2015-11_04

44|aaaa-bbb-ccc
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1 TERMINATED BY ' ',

C2,

C3 DATE FORMAT 'yyyy-mm-dd'

)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'

badfile=\'/opt/data/test.bad\'
```

5. 查看错误数据文件/opt/data/test.bad，其内容如下：

```
dmfldr: 2015-11-09 16:56:52 SYSDBA->TEST 4|4 aaaa-bbb-ccc
```

3.4 服务器端错误数据处理

dmfldr 客户端将载入的数据进行数据转换和编码转换后，将转换正确的数据发往 DM 服务器的 dmfldr 模块，也就是 dmfldr 的服务器端，由其进行真正的数据载入工作。

dmfldr 客户端每次向服务器端发送一批数据，在服务器端插入数据的过程中，由于目的表上可能存在约束等原因，导致某些数据无法插入成功，此时服务器端会将这一批数据全部回滚，并将这批数据全部记为错误数据，但服务器端插入时的错误数据并不会记录到 BADFILE 中。

ERRORS 所统计的错误包含在数据转换和数据插入过程中所产生的数据错误，因此当服务器端插入数据记录的错误数据数加上客户端数据转换时的错误数据数超过 ERRORS 参数的指定值时，dmfldr 服务器会停止插入数据。

3.5 大字段数据处理

dmfldr 支持对 DM 数据库的大字段类型数据的载入和导出，DM 数据库支持的大字段数据类型有 TEXT、LONGVARCHAR、IMAGE、LONGVARBINARY、BLOB 以及 CLOB。

3.5.1 大字段数据的导出

当 dmfldr 工作在导出模式即 MODE 为 OUT 时，dmfldr 生成大字段对应的数据文件名由 LOB_FILE_NAME 指定，若未指定默认为 dmfldr.lob，文件存放于 LOB_DIRECTORY 指定的目录，如果未指定 LOB_DIRECTORY 则存放于指定的导出数据文件同一目录。

例 展示大字段数据的导出。

1. 建表 TEST。

```
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 BLOB,C3 CLOB);
```

2. 插入数据。

```
INSERT INTO TEST VALUES(1,0XAB121032DE,'abcdefg');
INSERT INTO TEST VALUES(2,0XAB121032DE,'abcdefg');
COMMIT;
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
FIELDS '|'
(
C1,
C2,
C3
)
```

4. 使用 dmfldr 进行导出数据。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
LOB_DIRECTORY=\'/opt/data/\' mode=\'out\'
```

在这个例子中，指定了 LOB_DIRECTORY，而没有指定 LOB_FILE_NAME，导出的大字段数据文件将存放在 LOB_DIRECTORY 指定的/opt/data 目录，文件名为 dmfldr.lob。

3.5.2DIRECT 为 TRUE 时大字段数据的载入

当 MODE 为 IN 且 DIRECT 为 TRUE 时，此时数据载入若涉及到大字段对象，需要用户指定大字段数据文件。若 CLIENT_LOB 为 TRUE，LOB_DIRECTORY 应指定大字段数据文件所在的客户端本地目录，且此时待载入的大字段长度不得超过 2G，否则无法成功载入大字段；若 CLIENT_LOB 为 FALSE，用户必须先把相关文件传送到 DM 服务器所在主库，然后使用 LOB_DIRECTORY 指明存放目录。

大字段数据文件在数据文件中指定，可以是任意格式的文件。在数据文件中，大字段以“文件名：起始偏移：长度”的形式记录在数据文件中。指定的文件名无效时，dmfldr 会报错，装载失败。对于 CLOB 类型字段，当指定的偏移、长度范围内带有不完整字符时，dmfldr 将装载失败。

例 展示 DIRECT 为 TRUE 时大字段数据的载入。

1. 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT,C2 BLOB,C3 CLOB);
```

2.编辑数据文件 test.txt，存放路径为 DM 服务器所在主库的/opt/data/test.txt，文件内容如下：

```
1|testblob.txt:0:10|testclob.txt:0:10

2|testblob.txt:10:20|testclob.txt:10:20

3|testblob.txt:20:30|testclob.txt:20:30
```

其中，testblob.txt、testclob.txt 为文本文件，长度大于 30 字节，存放路径为/opt/data。

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1,

C2,

C3

)
```

4. 使用 dmfldr 进行导入数据。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'

LOB_DIRECTORY=\'/opt/data/\'
```

3.5.3DIRECT 为 FALSE 时大字段数据的载入

当 MODE 为 IN 且 DIRECT 为 FALSE 时，数据文件中大字段列数据即字段内容。此时，大字段数据长度不允许超过 32KB。BLOB_TYPE 参数指定 BLOB 列内容为十六进制或者字符串：

- BLOB_TYPE 为 HEX_CHAR 时，数据文件中 BLOB 列当作为十六进制内容；

- BLOB_TYPE 为 HEX 时，数据文件中 BLOB 列为字符串形式内容，导入后会转换为十六进制。

BLOB_TYPE 参数只对 DIRECT 为 FALSE 时有效，默认为 HEX_CHAR。

注意： **不支持使用包含BLOB列的表生成的导出数据文件在导入时指定BLOB_TYPE为HEX_CHAR。**

例 1 展示 DIRECT 为 FALSE 的大字段数据载入。

1. 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT,C2 BLOB,C3 CLOB);
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1|0x12d3c8a7|abcdefg

2|0x12a4cbac|hijlkmn

3|0x22d3c8b3|adefhjd
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1,

C2,

C3

)
```

4. 使用 dmfldr 进行导入数据，BLOB_TYPE 为 HEX_CHAR。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'

direct=false blob_type=\'hex_char\'
```

5. 查询表数据。

```
SELECT * FROM TEST;
```

```
行号        C1          C2          C3
---------- ----------- ---------- -------
1           1           0x12D3C8A7 abcdefg
2           2           0x12A4CBAC hijlkmn
3           3           0x22D3C8B3 adefhjd
```

例 2 展示 BLOB_TYPE 为 HEX 时的数据装载。

步骤 1、2、3 同例 1。

4．使用 dmfldr 进行导入数据，BLOB_TYPE 为 HEX。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
direct=false blob_type=\'hex\'
```

5． 查询表数据。

```
select * from test;
行号        C1          C2                      C3
---------- ----------- ---------------------- -------
1           1           0x30783132643363386137 abcdefg
2           2           0x3078313261346263163 hijlkmn
3           3           0x30783232643363386233 adefhjd
```

3.6 日志文件及日志信息

dmfldr 的日志文件路径由 LOG 参数设置，默认日志文件名为 fldr.log。文件记录了装载过程中的装载信息和错误信息以及统计信息。用户也可以通过设置控制文件中的 OPTIONS 选项来指定日志路径。如果参数及 OPTION 中同时指定了日志路径则其将以 OPTION 中指定的路径为最终路径。

例 展示日志文件的用法。

1． 建表 TEST。

```
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 INT,C3 DATE);
```

2． 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1|1 2018-11-06
```

```
2|2 2018-11-05

3|3 2018-11_04
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1,

C2 TERMINATED BY ' ',

C3 DATE FORMAT 'yyyy-mm-dd'

)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'

log=\'/opt/data/test.log\'
```

5. 查看日志文件/opt/data/test.log，其内容如下：

```
dmfldr: 2018-12-03 13:46:33  dmfldr:

Copyright (c) 2011, 2015, Dameng.  All rights reserved.

控制文件:

加载行数:全部

每次提交服务器行数:50000

跳过行数:0

允许错误数:100

是否直接加载:Yes

是否插入自增列:No

数据是否已按照聚集索引排序:No

字符集:GBK

dmfldr: 2018-12-03 13:46:33

数据文件共 1 个:

d:\test.txt
```

```
错误文件:fldr.bad
目标表:TEST
列名        包装数据类型          终止
C1          CHARACTER           |
C2          CHARACTER           WHT
C3          yyyy-mm-dd          |
行缓冲区数量: 4
任务线程数量: 4
dmfldr: 2018-12-03 13:46:33
目标表:TEST
3 行加载成功。
由于数据格式错误,0 行 丢弃。
由于数据错误,0 行 没有加载。
跳过的逻辑记录总数:0
读取的逻辑记录总数:3
拒绝的逻辑记录总数:0
用时:20.522(ms)
```

3.7 自增列装载

自增列是比较特殊的列，为了保证数据库中自增列列值的正确性，用户在进行数据载入时需要特别注意。

当 DIRECT 参数为 FALSE 时，dmfldr 将把从数据文件中读取的自增列值作为目标值插入数据库表中，用户应当保证每一行的自增列的值符合自增列的规则，否则将造成数据混乱。

当 DIRECT 参数为 TRUE 时，dmfldr 提供了 SET_IDENTITY 参数（缺省为 FALSE）对数据载入时自增列的处理进行设置：

- ✓ 如果指定 SET_IDENTITY 选项值为 TRUE，则 dmfldr 将把从数据文件中读取的自增列值作为目标值插入数据库表中，用户应当保证每一行的自增列的值符合自增列的规则，否则将造成数据混乱；

- ✓ 如果 SET_IDENTITY 选项值设置为 FALSE，则 dmfldr 将忽略数据文件中对应自增列的值，服务器将根据自增列定义和表中已有数据自动生成自增列的值插入每一行的对应列。

另外需要注意的是，DMDPC 环境下暂不支持自增列装载。

例 1 展示自增列的装载。

1. 建表 TEST，并插入两行数据。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT IDENTITY(1,1),C2 VARCHAR);

INSERT INTO TEST(C2) VALUES('AAA');

INSERT INTO TEST(C2) VALUES('BBB');

COMMIT;
```

此时表 TEST 中的数据为：

```
行号        C1           C2

---------- ----------- ---

1           1            AAA

2           2            BBB
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
2|aaa

3|bbb

4|ccc
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1,

C2

)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
direct=true set_identity=false
```

5． 查看表 TEST 的数据。

```
SELECT * FROM TEST;
行号        C1          C2
---------- ----------- ---
1           1           AAA
2           2           BBB
3           3           aaa
4           4           bbb
5           5           ccc
```

在这个例子中，表 TEST 中已有两行数据，由于 SET_IDENTITY 置为 FALSE，因此在数据载入时 dmfldr 根据 C1 列的定义和表中已有数据，为 C1 列重新插入合适的值。

我们再看看如果将 SET_IDENTITY 置为 TRUE 结果会怎样。

例 2 展示 SET_IDENTITY 置为 TRUE 时的装载。

重复例 1 的 1、2、3 步骤。

4． 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
direct=true set_identity=true
```

5． 查看表 TEST 的数据。

```
SELECT * FROM TEST;
行号        C1          C2
---------- ----------- ---
1           1           AAA
2           2           BBB
3           2           aaa
4           3           bbb
5           4           ccc
```

3.8 数据排序

SORTED 参数用来设置数据是否已经按照聚集索引排序，缺省为 FALSE。

如果设置为 TRUE，则用户必须保证数据已按照聚集索引排序完成，并且如果表中存在数据，需要插入的数据索引值要比表中数据的索引值大，服务器在做插入操作时顺序进行插入。若数据并未按照索引排序，则 dmfldr 会报错，装载失败。

如果设置为 FALSE，则服务器对于每条记录进行定位插入。

此参数仅在 MODE 为 IN 且 DIRECT 为 TRUE 的情况下有效。

在数据量大，并且确定数据已按照聚集索引排序完成的情况下，将 SORTED 参数设置为 TRUE，可以提升装载性能。

例 展示数据排序。

1． 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT CLUSTER PRIMARY KEY,C2 VARCHAR);
```

2． 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
2|aaa

3|bbb

4|ccc

5|ddd

1|zzz
```

3． 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test

FIELDS '|'

(

C1,

C2

)
```

4． 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
sorted=true
```

由于本例中数据文件中的数据并没有按照 C1 列排序，dmfldr 将会报错。

```
不能使用 NOSORT 选项，数据非有序
```

3.9 空值处理

dmfldr 通过设置 NULL_MODE 参数来处理空值。

- 设置为 TRUE，载入时 NULL 字符串处理为 NULL，载出时空值处理为 NULL 字符串
- 设置为 FALSE，载入时 NULL 字符串处理为字符串，载出时空值处理为空串

例 1 展示 null_mode 为 true 时的空值处理。

1． 建表 TEST。

```
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 VARCHAR);
```

2． 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1|aaa
2|NULL
3|null
```

3． 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
FIELDS '|'
(
C1,
C2
)
```

4． 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
null_mode=true
```

5.查看表 TEST 的数据，数据文件中 C2 列的“NULL”和“null”字符串都被处理为空值。

```
SELECT C1, IFNULL(C2, 'NULL VALUE') FROM TEST;

行号        C1           IFNULL(C2,'NULLVALUE')

---------- ----------- ----------------------

1           1            aaa

2           2            NULL VALUE

3           3            NULL VALUE
```

例 2 展示 null_mode 为 false 时的空值处理。

步骤 1、2、3 与例 1 相同。

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'

null_mode=false
```

5.查看表 TEST 的数据，数据文件中 C2 列的“NULL”和“null”字符串直接被作为字符串值插入表中。

```
SELECT C1, IFNULL(C2, 'NULL VALUE') FROM TEST;

行号        C1           IFNULL(C2,'NULLVALUE')

---------- ----------- ----------------------

1           1            aaa

2           2            NULL

3           3            null
```

3.10 类类型装载

dmfldr 支持对 CLASS 类型数据的装载。CLASS 类型装载和 LOB 类型一样，载入时需要指定 LOB_DIRECTORY，另外需要指定 ENABLE_CLASS_TYPE 为 TRUE；导出时，默认导出目录和数据文件所在目录一致，导出类类型形成的大字段文件默认为 dmfldr.lob，导出目录和大字段文件可通过 LOB_DIRECTORY 和 LOB_FILE_NAME 参数设置。

注意：CLASS 类型大字段数据文件无法通过手动创建，只有从表中导出到大字段文件中，并使用数据文件 CLASS 字段对应的偏移、字长进行导入。

例 展示类类型装载。

1. 建表 TEST。

数据准备，创建类 mycls 的类头与类体，之后创建表 TEST，其 C2 列类型为 mycls 类类型，并向 TEST 中插入两行数据。

```
//类头创建
CREATE CLASS mycls
AS
TYPE rec_type IS RECORD (c1 INT, c2 INT);      //类型声明
id     INT;                        //成员变量
r      rec_type;                   //成员变量
FUNCTION f1(a INT, b INT) RETURN rec_type;     //成员函数
FUNCTION mycls(id INT , r_c1 INT, r_c2 INT) RETURN mycls;//用户自定义构造函数
END;
/


//类体创建
CREATE OR REPLACE CLASS BODY mycls
AS
FUNCTION f1(a INT, b INT) RETURN rec_type
AS
BEGIN
       r.c1 = a;
       r.c2 = b;
       RETURN r;
END;
FUNCTION mycls(id INT, r_c1 INT, r_c2 INT) RETURN mycls
AS
```

```
BEGIN
       this.id = id;        //可以使用 this.来访问自身的成员
       r.c1 = r_c1;         //this 也可以省略
       r.c2 = r_c2;
       RETURN this;         //使用 return this 返回本对象
END;
END;
/


//建表 TEST
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 mycls);
//插入数据：
INSERT INTO TEST VALUES(1,mycls(1,1,1));
INSERT INTO TEST VALUES(2,mycls(2,2,2));
COMMIT;
```

2． 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
FIELDS '|'
(
C1,
C2
)
```

3． 使用 dmfldr 进行数据导出。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
mode=\'out\' lob_directory=\'/opt/data\'
```

4．在指定的 LOB_DIRECTORY 目录会生成大字段文件 dmfldr.lob，在指定的数据文件路径生成的数据文件内容如下：

```
1|dmfldr.lob:0:70

2|dmfldr.lob:70:70
```

5. 创建一张新表 TEST2，表的两列都为 mycls 类类型。

```
CREATE TABLE TEST2(C1 MYCLS,C2 MYCLS);
```

6. 编辑数据文件 test2.txt，存放路径为/opt/data/test2.txt，文件内容如下：

```
dmfldr.lob:70:70|dmfldr.lob:0:70

dmfldr.lob:0:70|dmfldr.lob:70:70
```

7. 编辑控制文件 test2.ctl，存放路径为/opt/data/test2.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test2.txt'

INTO TABLE test2

FIELDS '|'

(

C1,

C2

)
```

8. 使用 dmfldr 进行数据导入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test2.ctl\'

 lob_directory=\'/opt/data/\' enable_class_type=true
```

9. 查看表 TEST2 的数据

```
行号        C1                              C2

---------- ------------------------------ ------------------------------

1           SYSDBA.MYCLS(2,REC_TYPE(2,2)) SYSDBA.MYCLS(1,REC_TYPE(1,1))

2           SYSDBA.MYCLS(1,REC_TYPE(1,1)) SYSDBA.MYCLS(2,REC_TYPE(2,2))
```

3.11 条件过滤

通过在控制文件中指定 WHEN <field_conditions>子句，可以在装载过程中对数据进行过滤，符合 field_conditions 条件的数据才会被装载。

对于条件过滤的使用需注意以下几点：

- 判断条件中的操作符仅支持比较相等和不相等，即=、!=和<>这三个比较操作符；
- 目前仅支持使用 AND 连接多个过滤条件；
- BLANKS 和 WHITESPACE 表示若干个空格；
- 判断条件若使用(p1:p2)作为比较表达式，其意义与在 POSTION 子句中的意义相同，表示从该行指定位置获取数据进行比较，起始位置和结束位置表示的都是字节位置，包含边界 p1,p2；
- 如果判断条件中使用 colid 作为比较表达式，该列必须在 INTO 表的 coldef_option 中进行说明；
- 如果判断条件中使用 colid 作为比较表达式，判断条件中使用的列仅用于过滤，并没有对应表中的某个实际列，应在 col_def 中指明 FILLER 属性表示装载时跳过该列；
- 如果判断条件中比较数据是字符常量值，其长度小于比较表达式长度，则在其之后补充空格；如果判断条件中比较数据是二进制串常量，其长度小于比较表达式长度，则在之后补充 0。

例 展示条件过滤。

1. 建表 TEST。

```
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 INT);
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
12
23
32
48
91
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
WHEN C1 != '2'
(
```

```
C1 position (1:1),
C2 position (2:2)
)
```

4． 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
```

5． 查看表 TEST 的数据，可以看到数据文件中的行 2，3 被过滤掉了。

```
SELECT * FROM TEST;
行号        C1          C2
---------- ----------- -----------
1           1           2
2           3           2
3           4           8
4           9           1
```

3.12 多表装载

通过在控制文件中指定多个 INTO TABLE 子句，可以将一批数据同时向多个表进行装载。每个 INTO TABLE 子句中都可以指定 WHEN 过滤条件、FIELDS 子句和列定义子句。

对于多表装载的使用需注意以下几点：

- 对于第二个及其之后的 INTO TABLE 子句，在其 coldef_option 中，必须为所有列指定 POSITION 选项；
- 在 INTO TABLE 子句的目标表中，如果目标表各不相同，则一个批次就可以完成装载完成（不管含有多少表，只需扫描一次数据文件即可）；如果含有重复的目标表，则相同的表需要分批次导入（每当遇到相同的表时，就需要再扫描一遍源文件。N 个重复的表，就需要扫描 N 次源文件）。

例 展示多表装载。

1． 建表 TEST1、TEST2。

```
DROP TABLE TEST1;
DROP TABLE TEST2;
CREATE TABLE TEST1(C1 INT,C2 INT);
```

```
CREATE TABLE TEST2(C1 INT,C2 INT);
```

2. 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1,2

2,3

3,2

4,8

9,1
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA

INFILE '/opt/data/test.txt'

INTO TABLE test1

WHEN C1 != '1'

FIELDS ','

(

c1 position (1:1),

c2 position (3:3)

)

INTO TABLE test2

WHEN (3:3) = '2' AND c1 != '3'

FIELDS ','

(

c1 position (1:1),

c2 position (3:3)

)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
```

5. 查看表 TEST1 和 TEST2 的数据如下：

```
SELECT * FROM TEST1;

行号        C1          C2

---------- ----------- -----------
```

```
1           2           3

2           3           2

3           4           8

4           9           1

SELECT * FROM TEST2;

行号        C1          C2

---------- ----------- -----------

1           1           2
```

3.13 个性化设置

用户通过设置 dmfldr 的 SKIP、LOAD、ROWS 参数，可以根据自己的需求调整装载的起始行、装载最大行数以及每次提交的行数。

SKIP 参数用来设置跳过数据文件起始的逻辑行数，整型数值。默认的跳过起始行数为 0 行。如果用户指定了多个文件，且起始文件中的行数不足 SKIP 所指定的行数，则 dmfldr 工具会扫描下一个文件直至累加的行数等于 SKIP 所设置的行数或者所有文件都已扫描结束。

LOAD 参数用来设置装载的最大行数，整型数值。默认的最大装载行数为数据文件中的所有行数。LOAD 指定的值不包括 SKIP 指定的跳过的行数。

ROWS 参数用来设置每次提交的行数，整型数值。默认的提交行数为 50000 行。提交行数的值表示提交到服务器的行数，并不一定代表按照数据文件中的数据顺序的行数。用户可以根据实际情况调整每次提交的行数，以达到性能的最佳点。ROWS 参数作用于 MODE 为 IN 的情况下，当 MODE 为 OUT 时无效。

例 展示个性化设置。

1． 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT,C2 VARCHAR);
```

2． 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1|aaa

2|bbb
```

```
3|ccc
4|ddd
5|eee
6|fff
7|ggg
8|hhh
9|iii
10|jjj
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
FIELDS '|'
(
C1,
C2
)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
skip=3 load=5
```

5. 查看表 TEST 的数据，载入时跳过了数据文件的前 3 行数据，且只载入了 5 条数据。

```
SELECT * FROM TEST;
行号        C1           C2
---------- ----------- ---
1           4            ddd
2           5            eee
3           6            fff
4           7            ggg
5           8            hhh
```

3.14 主备切换时的数据继续载入

在 DM 数据守护主备环境下，如果 dmfldr 数据载入中发生主备切换，dmfldr 支持主备切换完成后自动继续装载数据，并且能保持数据正确。

要使用此项功能，USERID 参数需要使用主备服务名方式进行配置，例如：

```
dmfldr USERID=SYSDBA/SYSDBA@dw CONTROL='c:\fldr.ctl'
```

同时在 dm_svc.conf 配置文件中配置主备服务名，例如：

```
dw=(192.168.0.101:5236, 192.168.0.102:5236)
```

目前主备切换时的数据继续载入功能还存在以下功能限制：

- 目前仅支持单机的主备，不支持 MPP 主备。但是，若在 MPP 主备环境中使用 MPP_CLIENT=FALSE，等同于单机，也是支持的。
- 目前不支持分区表装载。

3.15MPP 本地分发

MPP_CLIENT 参数用来设置在 DM MPP 环境下使用 dmfldr 进行数据装载时的数据分发方式。

当 DM 数据库服务器环境为单站点时此参数无效。当服务器环境为 MPP 环境时，若 MPP_CLIENT 为 TRUE，dmfldr 采用客户端分发模式；若 MPP_CLIENT 为 FALSE，则采用本地分发模式。

客户端分发模式下，数据在 dmfldr 客户端分发好后直接往指定站点发送数据。

本地分发模式下，导入时，dmfldr 直接将数据发送到连接的站点，数据最终在连接的站点；导出时，dmfldr 只导出连接站点的数据。

MPP 环境下要配置 dmmal.ini 文件中的 MAL_INST_HOST 和 MAL_INST_PORT 参数。

例 当前 DM 数据库服务器环境为 MPP 环境，两个节点，SEQNO 号分别为 0、1。使用 dmfldr 采用本地分发方式进行数据装载，连接的节点为 SEQNO 为 0 的节点。

1． 建表 TEST。

```
DROP TABLE TEST;

CREATE TABLE TEST(C1 INT,C2 INT);
```

2． 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1|1
2|2
3|3
4|4
5|5
6|6
7|7
8|8
9|9
10|10
```

3. 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
FIELDS '|'
(
C1,
C2
)
```

4. 使用 dmfldr 进行数据载入。

```
./dmfldr userid=SYSDBA/SYSDBA@localhost:5236 control=\'/opt/data/test.ctl\'
mpp_client=false
```

5. 查询 SEQNO 为 0 的节点上 TEST 表的数据。

```
SELECT * FROM TEST WHERE SF_GET_EP_SEQNO(ROWID)=0;
行号        C1          C2
---------- ----------- -----------
1           1           1
2           2           2
3           3           3
4           4           4
```

```
5           5           5
6           6           6
7           7           7
8           8           8
9           9           9
10          10          10
```

6. 查询 SEQNO 为 1 的节点上 TEST 表的数据。

```
SELECT * FROM TEST WHERE SF_GET_EP_SEQNO(ROWID)=0;
未选定行
```

3.16 提升 dmfldr 性能

用户在使用 dmfldr 时根据系统和数据的具体情况对一些参数进行调整，可以获得更好的性能，这些参数包括：

- **BUFFER_NODE_SIZE**

BUFFER_NODE_SIZE 设置读取文件缓冲区页大小，值越大，缓冲区的页越大，每次读取的数据就越多，每次发送到服务器的数据也就越多，效率越高。但其大小受 dmfldr 客户端内存大小限制。

- **READ_ROWS**

在某些情况下，BUFFER_NODE_SIZE 读入的数据行数很大，而后续操作处理不了这么大的行数，此时可以用 READ_ROWS 来限制处理的行数。dmfldr 取 READ_ROWS 和 BUFFER_NODE_SIZE 中较小的值作为一次处理的行数。

- **SEND_NODE_NUMBER**

指定 dmfldr 在数据载入时发送节点的个数，默认由系统计算一个初始值。取值范围为 16~65535。若在数据载入时发现发送节点不够用，系统会动态增加分配。在系统内存足够的情况下，可以适当设大 SEND_NODE_NUMBER 值，提升 dmfldr 载入性能。

- **TASK_THREAD_NUMBER**

指定 dmfldr 在数据载入时处理用户数据的线程数目。默认情况下，dmfldr 将该参数值设为系统 CPU 的个数，但当 CPU 个数大于 8 时，缺省值都被置为 8。在 dmfldr 客户

端所在机器 CPU 大于 8 环境中，提高 TASK_THREAD_NUMBER 值可以提升 dmfldr 装载性能。

- **BLDR_NUM**

水平分区表装载时，指定服务器 BLDR 的最大个数，缺省为 64。

服务器的 BLDR 保存水平分区子表相关信息，BLDR_NUM 的设置也就指定了服务器能同时载入的水平分区子表的个数。若 BLDR_NUM 设置太大，当水平分区子表数过多时，可能会导致服务器内存不足。当载入时实际需要的 BLDR 个数超出 BLDR_NUM 设置时，会淘汰指定子表的 BLDR，并替换为新的子表 BLDR。

- **BDTA_SIZE**

BDTA（Batch Data）的大小，缺省为 5000。

BDTA 代表 DM 数据库批量数据处理机制中一个批量，在内存、CPU 允许的条件下，增大 BDTA_SIZE 能加快装载速度；在网络是装载性能瓶颈时，增大 BDTA_SIZE 影响不大。

- **INDEX_OPTION**

索引的设置选项，缺省为 1。INDEX_OPTION 的可选项有 1、2 和 3。

1 代表服务器装载数据时先不刷新二级索引，而是将新数据按照索引预先排序，在装载完成后，再将排好序的数据插入索引。如果在数据载入前，目标表中已有较多数据，建议 INDEX_OPTION 置为 1。

2 代表服务器在快速装载过程中不刷新二级索引数据，只在装载完成时重建所有二级索引。如果在数据载入前，目标表中没有数据或数据量较小，建议 INDEX_OPTION 置为 2。

3 代表服务器使用追加模式来进行二级索引的插入，在数据装载的过程中，同时进行二级索引的插入，若未禁用全局索引，则全局索引可通过设置 INDEX_OPTION 为 3 进行插入。当原有数据量远大于插入数据量时，建议 INDEX_OPTION 置为 3。

3.17dmfldr 使用限制

dmfldr 的使用存在以下一些限制：

- 不支持向临时表、外部表装载数据
- 不支持向系统表装载数据
- 不支持向带有位图索引的表装载数据
- 不支持向带有函数索引的表装载数据

- 不支持向带有全文索引的表装载数据
- 不支持向 DCP 代理装载数据
- dmfldr 装载时，对约束进行检查，对各种约束的处理机制如下表所示

表 3.1 dmfldr 的约束检查机制

约束	数据不满足时	数据插入与否	约束是否有效
非空约束(NOT NULL)	报错	不插入	有效
聚集索引(CLUSTER PRIMARY KEY)	报错	不插入	有效
唯一约束(UNIQUE， PRIMARY KEY)	报错	插入	失效
引用约束(FOREIGN KEY)	不报错	插入	有效
CHECK 约束(CHECK)	不报错	插入	有效

4 dmldrp 和 dmldrc 入门

dmldrp 和 dmldrc 只支持快速载入功能，不支持快速载出功能。

4.1 dmldrp

本章简单介绍如何启动轻量级快速装载工具服务器 dmldrp 及其支持的参数。通过阅读本章，读者可以了解 dmldrp 通过各参数能提供的各项功能。

4.1.1 启动 dmldrp

安装好 DM 数据库管理系统后，在安装目录“bin”子目录下可找到 dmldrp 执行文件。

启动操作系统的命令行窗口，进入 dmldrp 所在目录，可以准备启动 dmldrp 工具了。

dmldrp 的使用必须指定必要的参数，否则工具会报错“无效的参数个数”并退出。为 dmldrp 指定参数的格式为：

```
dmldrp keyword=value
```

例 启动一个端口号为 9898 的 dmldrp。

```
dmldrp.exe port=9898
```

4.1.2 查看 dmldrp 参数

用户可以使用“dmldrp help”查看 dmldrp 版本信息和各参数的简单信息。

```
dmldrp help

dmldrp V8

格式: dmldrp.exe  KEYWORD=value

例程: dmldrp.exe


关键字                说明

-------------------------------------------------------------------------

PORT                端口号(可选项, 默认为 8336)
```

```
HELP                    打印帮助信息
```

4.1.3dmldrp 参数简介

dmldrp 只有两个参数：端口号和 HELP。

- **PROT**

dmldrp 的端口号，用于 dmldrp 和 dmldrc 之间的通信连接。取值范围为 1024~65534。可选配置，不配置则使用默认的端口 8336。

- **HELP**

获取帮助信息。

4.2 dmldrc

本章简单介绍如何启动轻量级快速装载工具客户端 dmldrc 及其支持的参数。通过阅读本章，读者可以了解 dmldrc 通过各参数能提供的各项功能。

4.2.1 启动 dmldrc

安装好 DM 数据库管理系统后，在安装目录“bin”子目录下可找到 dmldrc 执行文件。

启动操作系统的命令行窗口，进入 dmldrc 所在目录，可以准备启动 dmldrc 工具了。

dmldrc 的使用必须指定必要的参数，否则工具会报错“无效的参数个数”并退出。为 dmldrc 指定参数的格式为：

```
dmldrc keyword=value [keyword=value ...]
```

例 dmldrc 连接一个端口号为 9898 的 dmldrp。

```
dmldrc userid=SYSDBA/SYSDBA@localhost:5236 localhost:9898

control=\'/opt/data/test.ctl\'
```

4.2.2 查看 dmldrc 参数

dmldrc 使用较为灵活，参数较多，用户可以使用“dmldrc help”查看 dmldrc 版本信息和各参数的简单信息。

```
dmldrc help

version: 03134283923-20220809-166764-10000

格式: dmldrc.exe   KEYWORD=value


例程: dmldrc.exe   SYSDBA/SYSDBA 192.168.0.1:8336


USERID 必须是命令行中的第一个参数

SERVER 必须是命令行中的第二个参数

字符串类型参数必须以引号封闭


关键字                说明（默认值）

------------------------------------------------------------------------------

---

USERID               用户名/口令， 格式:{<username>[/<password>] |

/}[@<connect_identifier>][<option>] [<os_auth>]

                     <connect_identifier> : [<svc_name> | host[:port] |

<unixsocket_file>]

                     <option> :

#{<exetend_option>=<value>[,<extend_option>=<value>]...}

                                  //此行外层{}是为了封装参数之用，书写时需要保留

                     <os_auth> : AS {SYSDBA|SYSSSO|SYSAUDITOR|USERS|AUTO}

SERVER               dmldrp 的 IP 地址和端口号。不需要指定参数名，直接指定 host[:port]

CONTROL              控制文件，字符串类型

LOG                  日志文件，字符串类型 (fldr.log)

BADFILE              错误数据记录文件，字符串类型 (fldr.bad)

SKIP                 初始忽略逻辑行数 (0)

LOAD                 需要装载的行数 (ALL)

ROWS                 提交频次 (50000)，DIRECT 为 FALSE 有效

DIRECT               是否使用快速方式装载 (TRUE)

SET_IDENTITY         是否插入自增列 (FALSE)
```

SORTED	数据是否已按照聚集索引排序 (FALSE)
INDEX_OPTION	索引选项 (1) 1 不刷新二级索引，数据按照索引先排序，装载完后再 将排序的数据插入索引 2 不刷新二级索引，数据装载完成后重建所有二级索引 3 刷新二级索引，数据装载的同时将数据插入二级索引
ERRORS	允许的最大数据错误数 (100)
CHARACTER_CODE	字符编码，字符串类型 (GBK, UTF-8, SINGLE_BYTE, EUC-KR)
MODE	装载方式，字符串类型 IN 表示载入，OUT 表示载出， OUTORA 表示载出 ORACLE (IN)
CLIENT_LOB	大字段目录是否在本地 (FALSE)
LOB_DIRECTORY	大字段数据文件存放目录
LOB_FILE_NAME	大字段数据文件名称，仅导出有效 (dmfldr.lob)
BUFFER_NODE_SIZE	读入文件缓冲区的大小 (10),有效值范围 1~2048
LOG_SIZE	日志信息缓冲区的大小 (1),有效值范围 1~100
READ_ROWS	工作线程一次最大处理的行数 (100000)，最大支持 2^26-10000
NULL_MODE	载入时 NULL 字符串是否处理为 NULL 载出时空值是否处理为 NULL 字符串 (FALSE)
NULL_STR	载入时视为 NULL 值处理的字符串
SEND_NODE_NUMBER	运行时发送节点的个数 (20)，有效值范围 16~65535
TASK_THREAD_NUMBER	处理用户数据的线程数目，默认与处理器核数量相同，有效值范围 1~128
BLDR_NUM	服务器 BLDR 数目 (64),有效值范围 1~1024
BDTA_SIZE	bdta 的大小 (5000)，有效值范围 100~10000
COMPRESS_FLAG	是否压缩 bdta (FALSE)
MPP_CLIENT	MPP 环境，是否本地分发 (TRUE)
SINGLE_FILE	MPP/DPC 环境，是否只生成单个数据文件(FALSE)
LAN_MODE	MPP/DPC 环境，是否以内网模式装载数据(FALSE)
UNREP_CHAR_MODE	非法字符处理选项(0),为 0 时表示跳过该数据行，为 1 时表示使用(*)替换错误字节
SILENT	是否静默方式装载数据(FALSE)

BLOB_TYPE	BLOB 类型字段数据值的实际类型，字符串类型 (HEX_CHAR) HEX 表示值为十六进制，HEX_CHAR 表示值为十六进制字符类型 仅在 direct=FALSE 有效
OCI_DIRECTORY	OCI 动态库所在的目录
DATA	指定数据文件路径
ENABLE_CLASS_TYPE	允许用户导入 CLASS 类型数据 (FALSE)
FLUSH_FLAG	提交时是否立即刷盘 (FALSE)
IGNORE_BATCH_ERRORS	是否忽略错误数据继续导入 (FALSE)
SINGLE_HLDR_HP	是否使用单个 HLDR 装载 HUGE 水平分区表 (TRUE)
EP	指定需要发送数据的站点序号列表，仅向 MPP/DPC 环境导入数据时有效
PARALLEL	是否开启并行装载(FALSE)
SQL	使用自定义查询语句，仅导出模式有效
TABLE	导入/出表
ROW_SEPERATOR	行分隔符
FIELD_SEPERATOR	列分隔符
COMMIT_OPTION	提交选项(0)，0:每发送一批数据后提交，1:发送完所有数据后提交
APPEND_OPTION	追加选项(0)，0：追加方式，1：替代方式，2：插入方式
COLNAME_HEADING	是否在导出文件头中打印列名(FALSE)
IGNORE_AIMLESS_DATA	是否忽略无目标数据(FALSE)
LOB_AS_VARCHAR	是否将 CLOB 作为 VARCHAR 进行导入导出(FALSE)
LOB_AS_VARCHAR_SIZE	将 CLOB 作为 VARCHAR 进行导入导出时，lob 数据最大大小(10)MB
LOG_LEVEL	记录错误数据信息级别(3)，0：不记录 1：只记录到 log 文件 2：只记录到 bad 文件 3：记录到 log 和 bad 文件
FLDR_INI	配置文件路径，字符串类型
RECONN	自动重连次数(0)
RECONN_TIME	自动重连等待时间(5)，单位(s)，有效值范围(1~10000)
HELP	打印帮助信息

4.2.3dmldrp 参数简介

dmldrp 参数除了 SERVER 以外，其它参数用法和 dmfldr 参数用法完全一样。本节只介绍 SERVER 参数用法，其它参数用法请参考 2.3dmfldr 参数简介。

- **SERVER**

用于指定 dmldrp 的 IP 地址和端口号。必选参数，且必须位于参数位置的第二个。

该参数不需要指定参数名 SERVER，直接指定 host[:port]。端口号缺省为 8336，本机 IP 可写做 localhost。

例 在 dmlrc 中指定 IP 和端口号分别为 192.168.1.64 和 9898 的 dmldrp。MODE 缺省为 IN。

```
dmldrc userid=SYSDBA/SYSDBA@localhost:5236  192.168.1.64:9898
control=\'/opt/data/test.ctl\'
```

5 dmldrp 和 dmldrc 实战

dmldrp 和 dmldrc 需搭配使用才能完成快速装载。具体分为两步：一是先启动 dmldrp；二是使用 dmldrc 执行导入功能。

例 使用 dmldrp 和 dmldrc 工具将数据快速装载进 test 表。

1.数据准备。

1） 建表 test。

```
DROP TABLE TEST;
CREATE TABLE TEST(C1 INT,C2 INT,C3 DATE);
```

2） 编辑数据文件 test.txt，存放路径为/opt/data/test.txt，文件内容如下：

```
1 1|2015-11-06
2 2|2015-11-05
3 3|2015-11_04
```

3） 编辑控制文件 test.ctl，存放路径为/opt/data/test.ctl，内容如下：

```
LOAD DATA
INFILE '/opt/data/test.txt'
INTO TABLE test
```

```
FIELDS '|'
(
C1 TERMINATED BY ' ',
C2,
C3 DATE FORMAT 'yyyy-mm-dd'
)
```

2.使用 dmldrp 和 dmldrc 完成快速装载。

1） 启动 dmldrp 服务器。

```
./dmldrp.exe port=9898
```

2） 使用 dmldrc 进行数据载入。

```
dmldrc userid=SYSDBA/SYSDBA@localhost:5236 localhost:9898
control=\'/opt/data/test.ctl\' MODE=\'IN\'
```

咨询热线：400-991-6599

技术支持：dmtech@dameng.com

官网网址：www.dameng.com

武汉达梦数据库股份有限公司

Wuhan Dameng Database Co.,Ltd.

地址：武汉市东湖新技术开发区高新大道999号未来科技大厦C3栋16—19层

16th-19th Floor, Future Tech Building C3, No.999 Gaoxin Road, Donghu New Tech Development Zone,Wuhan,Hubei Province,China

电话：(+86) 027-87588000　传真：(+86) 027-87588810