目录

1. 概述

1.1. 目的

本文旨在指导如何安装、使用 pgloader 工具完成从 MySQL 将数据库迁移到 openGauss。

1.2. Pgloader 介绍

Pgloader 是一个使用 COPY 命令实现迁移的 PostgreSQL 数据加载工具。

与仅使用 COPY 或者是\copy 以及使用外部数据包装器相比，其主要优势在于其事务行为，即保留一个单独的被拒绝数据文件，但继续尝试复制数据库中的良好数据，以此保证数据的完整性和一致性。正因为这种事务性行为，输入数据（如文件或远程数据库）中的任何错误行都将导致表的整个批量加载的停止。

除此之外，pgloader 还实现了数据重新格式化，如岂能将 MySQL 日期戳 00000-000 和 00000-000:00:00 转换为 PostgreSQL NULL 值（因为我们的日历从来没有年零）。

2. Pgloader 安装方法及打包教程

可以使用如下 git 命令获取源代码：

git clone https://gitee.com/opengauss/openGauss-tools-loader.git

2.1. 源代码安装

- 安装环境：Linux x86，freetds，epel，sbcl 1.2+，sqlite-devel，zlib-devel

如果可以使用 yum，可以通过 sh bootstrap-*.sh 来安装依赖，其中*代表的是当前的操作系统，如有 centos、centos7 以及 debian 三种。

- 离线安装：make pgloader
- 在线安装：make -f Makefile_online pgloader

2.2. 打包

- 环境要求：除安装教程的安装环境之外，还需要额外的 rpmdevel 依赖，在 bootstrap-*.sh 中会安装该依赖。
- 离线打包：make rpm
- 在线打包：make -f Makefile_online rpm

3. Pgloader 语法基础（MySQL 迁移）

命令示例如下所示（一般使用时不会这么复杂，后续有实例进行演示，此处仅作参考，尽量不要复制粘贴这里的命令来迁移）：

```
LOAD DATABASE
     FROM      mysql://root@localhost/sakila
     INTO postgresql://localhost:54393/sakila

 WITH include drop, create tables, create indexes, reset sequences,
      workers = 8, concurrency = 1,
      multiple readers per thread, rows per range = 50000

  SET PostgreSQL PARAMETERS
      maintenance_work_mem to '128MB',
      work_mem to '12MB',
      search_path to 'sakila, public, "$user"'

  SET MySQL PARAMETERS
      net_read_timeout  = '120',
      net_write_timeout = '120'

 CAST type bigint when (= precision 20) to bigserial drop typemod,
      type date drop not null drop default using zero-dates-to-null,
      -- type tinyint to boolean using tinyint-to-boolean,
      type year to integer

 MATERIALIZE VIEWS film_list, staff_list

 -- INCLUDING ONLY TABLE NAMES MATCHING ~/film/, 'actor'
 -- EXCLUDING TABLE NAMES MATCHING ~<ory>
 -- DECODING TABLE NAMES MATCHING ~/messed/, ~/encoding/ AS utf8
 -- ALTER TABLE NAMES MATCHING 'film' RENAME TO 'films'
 -- ALTER TABLE NAMES MATCHING ~/_list$/ SET SCHEMA 'mv'

 ALTER TABLE NAMES MATCHING ~/_list$/, 'sales_by_store', ~/sales_by/
```

```
  SET SCHEMA 'mv'

ALTER TABLE NAMES MATCHING 'film' RENAME TO 'films'
ALTER TABLE NAMES MATCHING ~/./ SET (fillfactor='40')

ALTER SCHEMA 'sakila' RENAME TO 'pagila'

BEFORE LOAD DO
   $$ create schema if not exists pagila; $$,
   $$ create schema if not exists mv;         $$,
   $$ alter database sakila set search_path to pagila, mv, public; $$;
```

下面对上述命令中的选项进行简要的介绍：

3.1. WITH 选项

从 MySQL 数据库加载时，支持以下选项。

默认的 WITH 子句为： no truncate, create tables, include drop, create indexes, reset sequences, foreign keys, downcase identifiers, uniquify index names

3.1.1.Include drop 及 include no drop

列出 Include drop 时，pgloader 将删除目标数据库中名称出现在 MySQL 数据库中的所有表。此选项允许连续多次使用同一命令，直到您找出所有选项，从干净的环境自动启动。请注意，CASCADE 用于确保表被删除，即使有外键指向它们。这正是包含删除的目的：删除所有目标表并重新创建它们。

使用包含删除时需要非常小心，因为它将级联到引用目标表的所有对象，可能包括未从源 DB 加载的其他表。

列出 include no drop 时，pgloader 在加载数据时将不包括任何删除语句。

3.1.2.Truncate 及 no truncate

列出 truncate 选项时，pgloader 在将数据加载到每个表之前，对每个表发出 TRUNCATE 命令。

3.1.3.disable triggers

列出此选项时，pgloader 会发出一个 ALTER TABLE...在复制数据之前，对目标表禁用触发 ALL 命令，然后使用命令 ALTER TABLE...复制完成后，启用触发 ALL。

此选项允许将数据加载到预先存在的表中，忽略外键约束和用户定义的触发器，并在加

载数据后可能导致无效的外键约束。小心使用。

3.1.4.create tables 及 create no tables

列出 create tables 选项时，pgloader 使用 MySQL 文件中找到的元数据创建表，该文件必须包含具有其数据类型的字段列表。完成了从 DBF 到 PostgreSQL 的标准数据类型转换。

当列出 create no tables 选项时，pgloader 在加载数据之前跳过表的创建，目标表必须已经存在。

此外，当使用 create no tables 时，pgloader 从当前目标数据库获取元数据并检查类型转换，然后将在加载数据之前删除约束和索引，并在加载完成后再次安装它们。

3.1.5.create indexes、create no indexes 及 drop indexes

列出 create indexes 选项时，pgloader 将获取 MySQL 数据库中找到的所有索引的定义，并针对数据库创建相同的索引定义集。若列出 create no indexes 选项则 pgloader 会跳过创建索引的阶段。

列出 drop indexes 选项时，pgloader 在加载数据之前删除目标数据库中的索引，并在数据副本结束时再次创建索引。

3.1.6.uniquify index names, preserve index names

MySQL 索引名称每个表是唯一的，而在 OpenGauss 中，索引名称必须是每个 schema 唯一的。pgloader 的默认值是通过在索引名称前面加上 idx_OID 来更改索引名称，其中 OID 是构建索引所依据的表的内部数字标识符。

在某些情况下，如 DDL 完全留给框架时，pgloader 可能避免处理索引唯一名称，这是通过使用保留索引名称选项来实现的。

默认值是唯一索引名称。即使使用保留索引名称选项，名为“PRIMARY”的 MySQL 主键索引也会获得唯一的名称。如果不这样做，将阻止在 OpenGauss 中再次创建主键，其中索引名称必须是每个 schema 唯一的。

3.1.7.drop schema

列出此选项时，pgloader 将在再次创建目标数据库及其包含的所有对象之前删除目标 schema。默认行为不会删除目标 schema。

3.1.8.foreign keys 及 no foreign keys

列出此选项时，pgloader 将获取 MySQL 数据库中找到的所有外键的定义，并针对数据库创建相同的外键定义集。否则，则跳过外键的创建。

3.1.9.reset sequences 及 reset no sequences

列出此选项时，在数据加载结束时以及所有索引创建后，pgloader 将创建的所有 OpenGauss 序列重置为它们附加到的列的当前最大值。否则则会跳过重置。

注 1：迁移至 openGauss 时请务必在迁移命令中写上 reset no sequences（因为默认值为 reset sequences），因为在进行序列重置时会使用到 openGauss 所不支持的 LISTEN 命令导致数据迁移失败。不使用序列重置时，在插入数据的过程中可能会产生序列号冲突的现象。因此，在对 MySQL 中有 AUTO INCREMENT 的表进行迁移后在请使用 setval 函数手动设置序列的起始值。

注 2：使用 schema only 和 data only 时该选项不会生效。

3.1.10. downcase identifiers

列出此选项时，pgloader 将所有 MySQL 标识符（表名、索引名、列名）转换为小写，openGauss 保留关键字除外。

openGauss 保留关键字是通过使用系统函数 pg_get_keywords()动态确定的。

3.1.11. quote identifiers

列出此选项时，pgloader 引用所有 MySQL 标识符，以便保留它们的大小写。

注：迁移后在应用代码中也要注意区分大小写。

3.1.12. schema only 及 data only

前者只迁移结构不迁移数据，后者则只发出复制数据的语句，不进行其他任何的处理。举个例子，当列出 schema only 的选项时，表中的索引会被保留下来，但索引内容为空，表中也没有数据；而列出 data only 则只有数据而没有索引。

3.1.13. single reader per thread, multiple readers per thread

默认值为每个线程单个读取器，这意味着每个 MySQL 表都由单个线程作为一个整体读取，单个 SELECT 语句不使用 WHERE 子句。

当每个线程使用多个读取器时，pgloader 可能能够将读取工作分为几个线程，与并发设置一样多，并发设置需要大于 1 才能激活此选项。

对于每个源表，pgloader 在单个数字列上搜索主键，或在第一列为数字数据类型（整数或 Bigint 之一）的多列主键索引上搜索主键。当存在这样的索引时，pgloader 运行查询以查找此列上的最小值和最大值，然后将该范围拆分为多个范围，每个范围最多包含行。

当我们然后获得的范围列表包含的范围至少与我们的并发设置一样多时，我们将这些范围分发到每个读取器线程。

因此，当满足所有条件时，pgloader 随后启动与并发设置一样多的读取器线程，每个读取器线程发出几个查询，其中 y - x =每个范围或更少的行数(对于最后一个范围，取决于刚刚

获得的最大值。

3.1.14. rows per range

此项列出当每个线程使用多个读取器时，每个 SELECT 查询读取多少行。

3.1.15. SET MySQL PARAMETERS

SET MySQL 参数允许在每次 pgloader 连接到 MySQL 时使用 MySQL SET 命令设置 MySQL 参数。

3.2. MySQL 数据库类型转换规则-guards

命令“CAST”引入用户定义的类型规则。

强制转换子句允许指定自定义强制转换规则，或者重载默认强制转换规则，或者使用特殊情况修改它们。

转换规则应遵循以下形式之一：

```
type <mysql-type-name> [ <guard> ... ] to <pgsql-type-name> [ <option> ... ]
column <table-name>.<column-name> [ <guards> ] to ...
```

强制转换规则可以与 MySQL 数据类型匹配，也可以与给定表名中的给定列名匹配。这种灵活性允许处理在某些情况下，tinyint 类型可能被用作布尔值，但在其他情况下，tinyint 类型可能被用作 smallint 情况。

强制转换规则按顺序应用，第一个匹配的转换规则会覆盖后续的规则，并首先评估用户定义的规则。

其中允许的 guards 如下所示：

3.2.1.when unsigned

强制转换规则仅适用于在其数据类型定义中具有关键字 unsigned 的源类型的 MySQL 列。以下是使用无符号保护的强制转换规则示例：

```
type smallint when unsigned to integer drop typemod
```

3.2.2.when default ‘value’

强制转换规则仅适用于具有给定值的源类型的 MySQL 列，该列必须是单引号或双引号字符串。

3.2.3.when typemod expression

强制转换规则仅适用于具有与给定 typemod 表达式匹配的 typemod 值的源类型的 MySQL 列。typemod 分为其精度和比例组件。

使用 typemod 防护的强制转换规则示例：

```
type char when (= precision 1) to char keep typemod
```

此表达式将 MySQL char(1)列强制转换为 char(1)类型的 PostgreSQL 列，同时允许默认强制转换规则将通用大小写 char(N)转换为 PostgreSQL 类型 varchar(N)。

3.2.4.with extra auto_increment

强制转换规则仅适用于设置了额外列 auto_increment 选项的 MySQL 列，因此可以针对串行而不是整数。

当未设置此选项时，默认匹配行为是将两个列与额外定义匹配，而不匹配。

这意味着，如果您要实现一个转换规则，该规则根据 MySQL 中的 auto_increment 额外信息位，以来自 smallint 定义的串行或整数为目标，则需要按以下方式说明两个转换规则：

```
type smallint  with extra auto_increment
 to serial drop typemod keep default keep not null,

type smallint
 to integer drop typemod keep default keep not null
```

3.3. MySQL 数据库类型转换规则-options

3.3.1.drop default, keep default

当列出 drop default 时，pgloader 将从其生成的 CREATE TABLE 语句中删除 MySQL 数据库中源类型列的任何现有默认表达式。

而 keep default 则是用于显式防止这种行为，并可用于重载默认强制转换规则。

3.3.2.drop not null, keep not null, set not null

当列出选项 drop not null 时，pgloader 在 PostgreSQL 数据库中创建表时，会删除与给定源 MySQL 数据类型关联的任何现有 NOT NULL 约束。

keep not null 显式防止这种行为，并可用于重载默认强制转换规则。

当列出 set not null 时，pgloader 在目标列上设置 NOT NULL 约束，而不管它是否已在源 MySQL 列中设置。

3.3.3.drop typemod, keep typemod

当列出选项 drop typemod，pgloader 在 PostgreSQL 数据库中创建表时，会从源类型的 MySQL 列中找到的数据类型定义中删除任何现有的 typemod 定义（例如精度和比例）。

keep typemod 显式防止这种行为，并可用于重载默认强制转换规则。

3.3.4.using

此选项将在 pgloader.transforms Common Lisp 包中找到的函数的名称作为其单个参数。

可以通过完全省略强制转换规则的类型部分，使用转换函数来扩展默认强制转换规则（例如适用于 ENUM 数据类型的强制转换规则），如以下示例所示：

```
column enumerate.foo using empty-string-to-null
```

3.4. 部分迁移

3.4.1.表名匹配迁移

引入逗号分隔的表名列表或正则表达式，用于限制要迁移到子列表的表。示例：

```
including only table names matching ~/film/, 'actor'
```

3.4.2.排除表名匹配

引入逗号分隔的表名列表或用于从迁移中排除表名的正则表达式。此过滤器仅适用于包括过滤器的结果。

```
excluding table names matching ~<ory>
```

3.5. MySQL 编码支持

3.5.1.解码表名匹配

引入逗号分隔的表名或正则表达式列表，用于强制在处理 MySQL 中的数据时使用编码。如果您已知的数据编码与 MySQL 关于它的想法不同，则可以使用此选项。

```
decoding table names matching ~/messed/, ~/encoding/ AS utf8
```

您可以根据需要使用尽可能多的此类规则，所有这些规则都可能具有不同的编码。

3.6. MySQL 模式转换

3.6.1. ALTER TABLE NAMES MATCHING

在 pgloader ALTER TABLE 命令中引入要针对的表名或正则表达式的逗号分隔列表。可用的操作包括 SET SCHEMA、RENAME TO 为和 SET：

```
ALTER TABLE NAMES MATCHING ~/_list$/, 'sales_by_store', ~/sales_by/
 SET SCHEMA 'mv'

ALTER TABLE NAMES MATCHING 'film' RENAME TO 'films'

ALTER TABLE NAMES MATCHING ~/./ SET (fillfactor='40')

ALTER TABLE NAMES MATCHING ~/./ SET TABLESPACE 'pg_default'
```

您可以根据需要使用尽可能多的此类规则。要迁移的表列表将根据 ALTER TABLE 匹配规则在 pgloader 内存中搜索，对于每个命令，pgloader 将停止在第一个匹配条件（regexp 或字符串）。

不会将 ALTER TABLE 命令发送到 PostgreSQL，修改发生在源数据库架构的 pgloader 内存表示的级别。如果名称更改，映射将保留并重用在外键和索引支持中。

SET ()操作作为 CREATE TABLE 命令的 WITH 子句生效，pgloader 在必须创建表时将运行该命令。

SET TABLESPACE 操作作为 pgloader 在必须创建表时将运行的 CREATE TABLE 命令的 TABLESPACE 子句生效。

4. 迁移限制

- 视图不迁移，支持视图可能需要使用移植引擎为 MySQL 方言实现完整的 SQL 解析器，以根据 PostgreSQL 重写 SQL，包括重命名函数和更改某些构造。
- 触发器未迁移。
- 在几何数据类型中，仅涵盖了点数据库。

5. 默认的类型转换规则

5.1. Numbers

```
type int with extra auto_increment to serial when (< precision 10)
type int with extra auto_increment to bigserial when (<= 10 precision)
```

```
type int to int              when    (< precision 10)
type int to bigint       when    (<= 10 precision)
type tinyint      with extra auto_increment to serial
type smallint     with extra auto_increment to serial
type mediumint with extra auto_increment to serial
type bigint       with extra auto_increment to bigserial

type tinyint to boolean when (= 1 precision) using tinyint-to-boolean

type bit when (= 1 precision) to boolean drop typemod using bits-to-boolean
type bit to bit drop typemod using bits-to-hex-bitstring

type bigint when signed to bigint drop typemod
type bigint when (< 19 precision) to numeric drop typemod

type tinyint when unsigned to smallint      drop typemod
type smallint when unsigned to integer    drop typemod
type mediumint when unsigned to integer    drop typemod
type integer when unsigned to bigint       drop typemod

type tinyint to smallint      drop typemod
type smallint to smallint    drop typemod
type mediumint to integer    drop typemod
type integer to integer       drop typemod
type bigint to bigint           drop typemod

type float to float              drop typemod
type double to double precision drop typemod

type numeric to numeric keep typemod
type decimal to decimal keep typemod
```

5.2. Texts

```
type char             to char keep typemod using remove-null-characters
type varchar       to varchar keep typemod using remove-null-characters
type tinytext      to text using remove-null-characters
type text             to text using remove-null-characters
type mediumtext to text using remove-null-characters
type longtext      to text using remove-null-characters
```

5.3. Binary

```
type binary          to bytea using byte-vecotr-to-bytea
type varbinary     to bytea using byte-vecotr-to-bytea
type tinyblob       to bytea using byte-vecotr-to-bytea
type blob             to bytea using byte-vecotr-to-bytea
type mediumblob to bytea using byte-vecotr-to-bytea
type longblob       to bytea using byte-vecotr-to-bytea
```

5.4. Date

```
type datetime when default "0000-00-00 00:00:00" and not null
    to timestamptz drop not null drop default
          using zero-dates-to-null

type datetime when default "0000-00-00 00:00:00"
    to timestamptz drop default
          using zero-dates-to-null

type datetime with extra on update current timestamp when not null
    to timestamptz drop not null drop default
         using zero-dates-to-null

type datetime with extra on update current timestamp
    to timestamptz drop default
         using zero-dates-to-null

type timestamp when default "0000-00-00 00:00:00" and not null
    to timestamptz drop not null drop default
          using zero-dates-to-null

type timestamp when default "0000-00-00 00:00:00"
    to timestamptz drop default
          using zero-dates-to-null

type date when default "0000-00-00" to date drop default
          using zero-dates-to-null

type date to date
type datetime to timestamptz
type timestamp to timestamptz
type year to integer drop typemod
```

5.5. Geometric

type geometry to point using convert-mysql-point type point to point using convert-mysql-point type linestring to path using convert-mysql-linestring

5.6. Enums

枚举类型在 MySQL 中内联声明，在 PostgreSQL 中单独使用 CREATE TYPE 命令声明，因此 Enum Type 的每一列都将转换为以相同顺序使用相同标签定义的表和列名命名的类型。

如果源类型定义在默认强制转换规则中和命令中提供的强制转换规则中都不匹配，则使用带有 typemod 的类型名称。

6. 实例

以源代码安装为例，首先安装 pgloader，安装及验证方式如下所示：

```
[luozihao@ctupopenga00017 openGauss-tools-loader]$ make pgloader
tar -zxf build/offline_packge.tar.gz -C build/
sbcl --noinform --no-sysinit --no-userinit --load build/quicklisp.lisp                          \
             --load src/getenv.lisp                                                  \
             --eval '(quit)'

  ==== quicklisp quickstart 2015-01-28 loaded ====

    To continue with installation, evaluate: (quicklisp-quickstart:install)

    For installation options, evaluate: (quicklisp-quickstart:help)

sbcl --noinform --no-sysinit --no-userinit --load build/quicklisp/setup.lisp                   \
             --eval '(push :pgloader-image *features*)'                 \
             --eval '(setf *print-circle* t *print-pretty* t)'          \
             --eval '(push "/data/luozihao/pgloader_test/openGauss-tools-loader/" ql:*local-project-directories*)' \
             --eval '(ql:quickload "pgloader")'                         \
             --eval '(quit)'
To load "pgloader":
  Load 1 ASDF system:
    pgloader
; Loading "pgloader"
```

```
[luozihao@ctupopenga00017 openGauss-tools-loader]$ pgloader --version
pgloader version "3.6.2"
compiled with SBCL 1.4.0-1.el7
[luozihao@ctupopenga00017 openGauss-tools-loader]$
```

事先在 mysql 数据库中准备一份测试用的数据以及可以读取这些数据的用户。如下所示（部分数据）：

```
MariaDB [ambari]> show tables;
+------------------------------+
| Tables_in_ambari             |
+------------------------------+
| ClusterHostMapping           |
| QRTZ_BLOB_TRIGGERS           |
| QRTZ_CALENDARS               |
| QRTZ_CRON_TRIGGERS           |
| QRTZ_FIRED_TRIGGERS          |
| QRTZ_JOB_DETAILS             |
| QRTZ_LOCKS                   |
| QRTZ_PAUSED_TRIGGER_GRPS     |
| QRTZ_SCHEDULER_STATE         |
```

在 openGauss 数据库中创建一个数据库 test_ambari1，如下所示：

```
lzh_test=# create database ambari_test1 dbcompatibility 'B';
CREATE DATABASE
lzh_test=#
```

注意，这里的兼容类型 dbcompatibility 一定要加上，不然默认的兼容类型（即类型 A）不一定能够迁移成功。因为 MySQL 数据库时能够插入空字符串的，而兼容类型 A 则会将空字符串视为 NULL，假如该数据插入了有 NOT NULL 约束的列，就会导致迁移失败。

生成一个 pg.loader 文件，根据第三节的内容写入迁移的配置，如下所示：

```
LOAD DATABASE
     FROM      mysql://luozihao:openGauss@@123@[redacted]/ambari
     INTO postgresql://lzh:openGauss@@123@[redacted]:54323/test_ambari1

 WITH include drop, create tables, create indexes, reset no sequences,
      workers = 8, concurrency = 1,
      multiple readers per thread, rows per range = 50000

 CAST type tinyint to smallint drop typemod
;
```

注意，如果你的数据库密码中有:或@这两个特殊字符，则要对应增加一个:或@，否则 pgloader 会无法识别，如上述所示，我的密码是 openGauss@123，所以在配置文件中我的数据库密码应该写成 openGauss@@123。

最后，执行 pgloader pg.loader 就可以进行迁移了，迁移成功的效果如下所示：

```
2021-04-17T08:12:25.113000Z LOG pgloader version "3.6.3~devel"
2021-04-17T08:12:25.750000Z LOG Migrating from #<MYSQL-CONNECTION mysql://luozihao@[redacted]:3306/ambari {1006A0E733}>
2021-04-17T08:12:25.750000Z LOG Migrating into #<PGSQL-CONNECTION pgsql://lzh@[redacted]:54323/test_ambari1 {1006A0FEA3}>
2021-04-17T08:14:15.022000Z LOG report summary reset
                  table name     errors       rows      bytes      total time
---------------------------------  ---------  ---------  ---------  --------------
                fetch meta data          0        484                     0.362s
                  Create Schemas         0          0                     0.260s
                Create SQL Types         0          0                     0.142s
                   Create tables         0        222                    44.709s
                  Set Table OIDs         0        111                     0.095s
---------------------------------  ---------  ---------  ---------  --------------
 ambari.permission_roleauthorization         0        189     4.8 kB          0.703s
           ambari.host_role_command          0        227     4.1 MB          1.157s
               ambari.alert_history          0        137    26.7 kB          0.700s
           ambari.execution_command          0        227     2.1 MB          1.046s
              ambari.alert_grouping          0         77     0.5 kB          0.843s
                      ambari.stage           0         82   118.3 kB          1.159s
               ambari.clusterconfig          0         91   261.2 kB          1.053s
                     ambari.widget           0         56    39.8 kB          1.094s
                    ambari.request           0         62    14.7 kB          1.107s
    ambari.widget_layout_user_widget         0         56     0.5 kB          1.321s
               ambari.alert_current          0         66     6.3 kB          1.425s
       ambari.requestresourcefilter          0         50     2.3 kB          1.665s
    ambari.hostcomponentdesiredstate         0         26     1.3 kB          1.644s
```