恒生电子股份有限公司

MySQL 到 LightDB
迁移实施手册

恒生研究院

2022 年 4 月

文档修改记录

版本	修订人	修订说明	批准人	发布日期
1.0.0.0		初稿		20220101
1.0.0.1		v22.1 发版更新		20220406
1.0.0.1		v22.3 发版更新		20221108

说　明

模板版本信息
编辑部门：EPG
批准日期：2018/9/26

目 录

一、引言

1.1 编写目的

本文档为恒生电子股份有限公司 LightDB ltloader 迁移实施手册说明书，本文档主要阐述 LightDB ltloader 迁移实施的详细功能介绍，完整的数据库功能请参考《LightDB 用户手册》。LightDB ltloader 迁移实施手册提供了一种异构数据库 MySQL 向 LightDB 迁移方案，旨在满足数据库用户的需求，丰富了 LightDB 数据库的周边工具和使用。

1.2 预期读者

本文档主要适用于LightDB数据库的：

- 数据库管理员
- 开发工程师
- 测试工程师
- 技术支持工程师

1.3 参考文献

《LightDB 数据库安装手册》

《LightDB EM Installer 手册》

二、导入前目标数据库优化参数配置

```
maintenance_work_mem = 2048MB
work_mem = 256MB
temp_buffers=512MB
min_parallel_table_scan_size=8MB
min_parallel_index_scan_size=8MB
#max_stack_depth=80MB
lock_timeout=1800000
autovacuum_max_workers=10
```

三、ltloader 安装

ltloader 工具可以实现 MySQL 到 LightDB 的迁移，支持迁移表结构、索引、外键约束和数据。这里介绍使用 ltloader 工具来迁移 MySQL 数据到 LightDB 的步骤及注意事项。
https://pgloader.readthedocs.io/en/latest/ref/mysql.html

ltloader 工具随 LightDB 自动安装，命令默认在$LTHOME/bin 下

```
$ echo $LTHOME/bin/ltloader
/oracle/lightdb/base/lightdb-x/13.8-22.3/bin/ltloader
```

1.4 配置迁移文件文件 mysql2lightdb.conf

示例环境如下：
MySQL：地址 10.20.137.41;源库名: benchmarksql ; 用户名: root 密码: 123456
LightDB: 地址 10.20.30.199;目标库名称:benchmarksql（需要预先建好）; 用户名:benchmarksql
下面为示例的配置文件 mysql2lightdb.conf，特别说明如果密码带@字符需要写入两个@@进行转义处理；

创建配置文件后，直接 ltloader 调用即可，如果同一服务器安装多个版本数据库，需要使用与数据库同版本 ltloader 进行执行
vi mysql2lightdb.conf

```
LOAD DATABASE
FROM mysql://root:123456@10.20.137.41:5757/benchmarksql
INTO postgresql://benchmarksql:benchmarksql@10.20.30.199:5431/benchmarksql
WITH include drop, create tables, create indexes, reset sequences, workers=10,
concurrency=20, max parallel create index =20, foreign keys, prefetch rows =
100000, batch rows = 100000

CAST type int with extra auto_increment to serial drop typemod,
type int to int drop typemod,
type tinyint with extra auto_increment to serial drop typemod,
type smallint with extra auto_increment to serial drop typemod,
type mediumint with extra auto_increment to serial drop typemod,
type bigint with extra auto_increment to bigserial drop typemod,
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 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 numeric,
type double to numeric,
type numeric to numeric keep typemod,
type decimal to decimal keep typemod,
type datetime to timestamp drop default drop not null using zero-dates-to-null
BEFORE LOAD DO
$$ select 1; $$;
```

1.5 执行 ltloader mysql2lightdb.conf 进行迁移

执行以下命令（上述迁移文件，因为配置的 LightDB 用户访问只能本地访问，需要在迁移的 LightDB 主机执行），mysql2lightdb.conf 为配置文件名，然后就会根据配置的策略迁移表结构和数据，ltloader 命令在安装 LightDB 后可直接使用。
ltloader mysql2lightdb.conf

```
$ ltloader mysql2lightdb.conf
2022-10-25T10:14:44.158000+08:00 LOG Migrating from #<MYSQL-CONNECTION
mysql://root@10.20.137.41:5757/benchmarksql {100822A103}>
2022-10-25T10:14:44.158000+08:00 LOG Migrating into #<PGSQL-CONNECTION
pgsql://benchmarksql@10.20.30.199:5431/benchmarksql {100822B693}>
2022-10-25T10:15:59.023000+08:00 ERROR PostgreSQL Database error 40P01: deadlock
detected
DETAIL: Process 220572 waits for ShareUpdateExclusiveLock on relation 30482 of
database 30477; blocked by process 220573.
Process 220573 waits for ShareUpdateExclusiveLock on relation 30482 of database
30477; blocked by process 220572.
HINT: See server log for query details.
QUERY: CREATE INDEX idx_30482_c_w_id ON benchmarksql.bmsql_customer (c_w_id,
c_d_id, c_last, c_first);
2022-10-25T10:16:18.243000+08:00 ERROR PostgreSQL Database error 40P01: deadlock
detected
DETAIL: Process 221345 waits for ShareUpdateExclusiveLock on relation 30491 of
database 30477; blocked by process 221344.
Process 221344 waits for ShareUpdateExclusiveLock on relation 30491 of database
30477; blocked by process 221345.
HINT: See server log for query details.
QUERY: CREATE UNIQUE INDEX idx_30491_primary ON benchmarksql.bmsql_history
(hist_id);
2022-10-25T10:16:18.467000+08:00 ERROR PostgreSQL Database error 40P01: deadlock
detected
```

```
DETAIL: Process 221343 waits for ShareUpdateExclusiveLock on relation 30491 of database 30477; blocked by process 221344.
Process 221344 waits for ShareUpdateExclusiveLock on relation 30491 of database 30477; blocked by process 221343.
HINT: See server log for query details.
QUERY: CREATE INDEX idx_30491_h_customer_fkey ON benchmarksql.bmsql_history (h_c_w_id, h_c_d_id, h_c_id);
2022-10-25T10:16:44.221000+08:00 ERROR PostgreSQL Database error 40P01: deadlock detected
DETAIL: Process 222095 waits for ShareUpdateExclusiveLock on relation 30500 of database 30477; blocked by process 222097.
Process 222097 waits for ShareUpdateExclusiveLock on relation 30500 of database 30477; blocked by process 222095.
HINT: See server log for query details.
QUERY: CREATE INDEX idx_30500_o_customer_fkey ON benchmarksql.bmsql_oorder (o_w_id, o_d_id, o_c_id);
2022-10-25T10:16:44.433000+08:00 ERROR PostgreSQL Database error 40P01: deadlock detected
DETAIL: Process 222098 waits for ShareUpdateExclusiveLock on relation 30500 of database 30477; blocked by process 222097.
Process 222097 waits for ShareUpdateExclusiveLock on relation 30500 of database 30477; blocked by process 222098.
HINT: See server log for query details.
QUERY: CREATE INDEX idx_30500_o_w_id ON benchmarksql.bmsql_oorder (o_w_id, o_d_id, o_carrier_id, o_id);
2022-10-25T10:20:41.207000+08:00 ERROR PostgreSQL Database error 40P01: deadlock detected
DETAIL: Process 228198 waits for ShareUpdateExclusiveLock on relation 30503 of database 30477; blocked by process 228197.
Process 228197 waits for ShareUpdateExclusiveLock on relation 30503 of database 30477; blocked by process 228198.
HINT: See server log for query details.
QUERY: CREATE UNIQUE INDEX idx_30503_primary ON benchmarksql.bmsql_order_line (ol_w_id, ol_d_id, ol_o_id, ol_number);
2022-10-25T10:22:42.808000+08:00 ERROR PostgreSQL Database error 40P01: deadlock detected
DETAIL: Process 231684 waits for ShareUpdateExclusiveLock on relation 30506 of database 30477; blocked by process 231683.
Process 231683 waits for ShareUpdateExclusiveLock on relation 30506 of database 30477; blocked by process 231684.
HINT: See server log for query details.
QUERY: CREATE INDEX idx_30506_s_w_id ON benchmarksql.bmsql_stock (s_w_id, s_quantity);
```

```
2022-10-25T10:22:43.163000+08:00 ERROR PostgreSQL Database error 40P01: deadlock detected
DETAIL: Process 231682 waits for ShareUpdateExclusiveLock on relation 30506 of database 30477; blocked by process 231683.
Process 231683 waits for ShareUpdateExclusiveLock on relation 30506 of database 30477; blocked by process 231682.
HINT: See server log for query details.
QUERY: CREATE UNIQUE INDEX idx_30506_primary ON benchmarksql.bmsql_stock (s_w_id, s_i_id);
2022-10-25T10:22:45.180000+08:00 ERROR PostgreSQL Database error 42704: index "idx_30491_primary" does not exist
QUERY: ALTER TABLE benchmarksql.bmsql_history ADD PRIMARY KEY USING INDEX idx_30491_primary;
2022-10-25T10:22:45.182000+08:00 ERROR PostgreSQL Database error 42704: index "idx_30503_primary" does not exist
QUERY: ALTER TABLE benchmarksql.bmsql_order_line ADD PRIMARY KEY USING INDEX idx_30503_primary;
2022-10-25T10:22:45.182000+08:00 ERROR PostgreSQL Database error 42704: index "idx_30506_primary" does not exist
QUERY: ALTER TABLE benchmarksql.bmsql_stock ADD PRIMARY KEY USING INDEX idx_30506_primary;
2022-10-25T10:23:15.538000+08:00 ERROR PostgreSQL Database error 42830: there is no unique constraint matching given keys for referenced table "bmsql_stock"
QUERY: ALTER TABLE benchmarksql.bmsql_order_line ADD CONSTRAINT ol_stock_fkey FOREIGN KEY(ol_supply_w_id,ol_i_id) REFERENCES benchmarksql.bmsql_stock(s_w_id,s_i_id) ON UPDATE RESTRICT ON DELETE RESTRICT
2022-10-25T10:23:20.021000+08:00 LOG report summary reset
                table name    errors      rows     bytes     total time
------------------------------  ---------  ---------  ---------  --------------
               before load         0         1                  0.008s
            fetch meta data         0        38                  0.131s
             Create Schemas         0         0                  0.003s
           Create SQL Types         0         0                  0.004s
             Create tables         0        20                  0.033s
             Set Table OIDs         0        10                  0.005s
------------------------------  ---------  ---------  ---------  --------------
   benchmarksql.bmsql_config         0         4     0.1 kB         0.061s
 benchmarksql.bmsql_customer         0   3000000     1.6 GB      1m10.411s
 benchmarksql.bmsql_district         0      1000    93.5 kB         0.062s
  benchmarksql.bmsql_history         0   3000000   192.6 MB        21.912s
     benchmarksql.bmsql_item         0    100000     7.2 MB         1.543s
benchmarksql.bmsql_new_order         0    900000     8.6 MB         5.162s
   benchmarksql.bmsql_oorder         0   3000000   117.9 MB        19.498s
benchmarksql.bmsql_order_line         0  30008779     1.9 GB      3m43.462s
```

```
    benchmarksql.bmsql_stock          0   10000000     2.9 GB       2m9.232s
 benchmarksql.bmsql_warehouse         0        100     8.7 kB         0.047s
----------------------------  ---------  ---------  ---------  --------------
     COPY Threads Completion          0         10                7m51.222s
             Create Indexes          8         10               1m17.225s
      Index Build Completion          0         18                   9.511s
            Reset Sequences          0          0                  0.013s
               Primary Keys          3          7                  0.005s
        Create Foreign Keys          1          9                  34.836s
            Create Triggers          0          0                  0.000s
           Install Comments          0          0                  0.000s
----------------------------  ---------  ---------  ---------  --------------
          Total import time          ✓   50009883     6.7 GB       9m52.812s
```

如上会出现索引冲突报错，其中 8 个索引、3 个主键、1 个外键创建失败，需要将其单独整理后在目标端 LightDB 下的 benchmarksql 库进行执行

```
CREATE INDEX idx_30482_c_w_id ON benchmarksql.bmsql_customer (c_w_id, c_d_id,
c_last, c_first);
CREATE UNIQUE INDEX idx_30491_primary ON benchmarksql.bmsql_history (hist_id);
CREATE INDEX idx_30491_h_customer_fkey ON benchmarksql.bmsql_history (h_c_w_id,
h_c_d_id, h_c_id);
CREATE INDEX idx_30500_o_customer_fkey ON benchmarksql.bmsql_oorder (o_w_id,
o_d_id, o_c_id);
CREATE INDEX idx_30500_o_w_id ON benchmarksql.bmsql_oorder (o_w_id, o_d_id,
o_carrier_id, o_id);
CREATE UNIQUE INDEX idx_30503_primary ON benchmarksql.bmsql_order_line (ol_w_id,
ol_d_id, ol_o_id, ol_number);
CREATE INDEX idx_30506_s_w_id ON benchmarksql.bmsql_stock (s_w_id, s_quantity);
CREATE UNIQUE INDEX idx_30506_primary ON benchmarksql.bmsql_stock (s_w_id,
s_i_id);
ALTER TABLE benchmarksql.bmsql_history ADD PRIMARY KEY USING INDEX
idx_30491_primary;
ALTER TABLE benchmarksql.bmsql_order_line ADD PRIMARY KEY USING INDEX
idx_30503_primary;
ALTER TABLE benchmarksql.bmsql_stock ADD PRIMARY KEY USING INDEX
idx_30506_primary;
ALTER TABLE benchmarksql.bmsql_order_line ADD CONSTRAINT ol_stock_fkey FOREIGN
KEY(ol_supply_w_id,ol_i_id) REFERENCES benchmarksql.bmsql_stock(s_w_id,s_i_id)
ON UPDATE RESTRICT ON DELETE RESTRICT;
```

执行完成后，登录 lightdb 数据库中，修改目的端库（上述例子中即 lightdb 的 benchmarksql 库）的 search_path 为默认值（ltloader 在迁移数据过程中会修改目的端库的 search_path 配置项），执行以下 SQL 恢复 search_path 默认值：

alter database benchmarksql set search_path = default;

然后重新建立与 lightdb benchmarksql 库的会话连接。

至此，数据迁移完成。

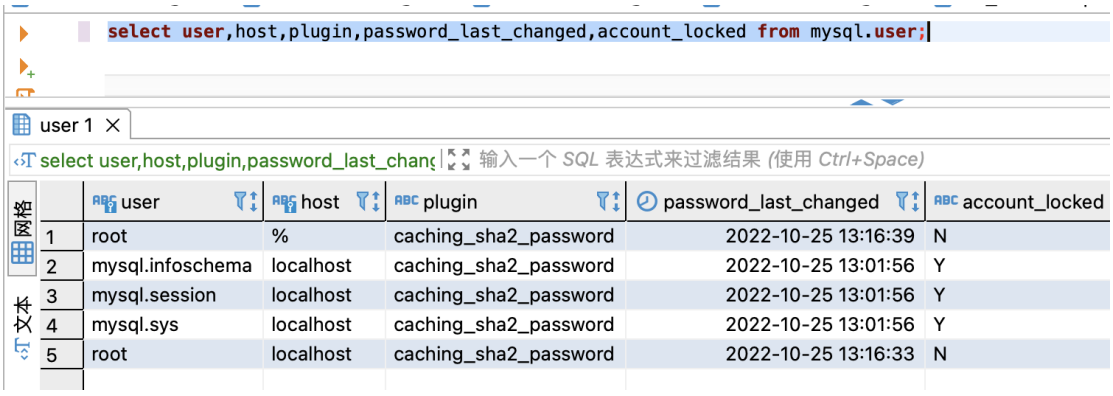
1.6 源端为 MySQL 8.0 迁移解决方案

如果认证方式为 sha256，会报如下错误

```
KABOOM!
FATAL error: Failed to connect to mysql at "10.20.137.41" (port 8800) as user
"root": Condition QMYND:MYSQL-UNSUPPORTED-AUTHENTICATION was signalled.
An unhandled error condition has been signalled:
   Failed to connect to mysql at "10.20.137.41" (port 8800) as user "root":
Condition QMYND:MYSQL-UNSUPPORTED-AUTHENTICATION was signalled.




What I am doing here?

Failed to connect to mysql at "10.20.137.41" (port 8800) as user "root": Condition
QMYND:MYSQL-UNSUPPORTED-AUTHENTICATION was signalled.
```

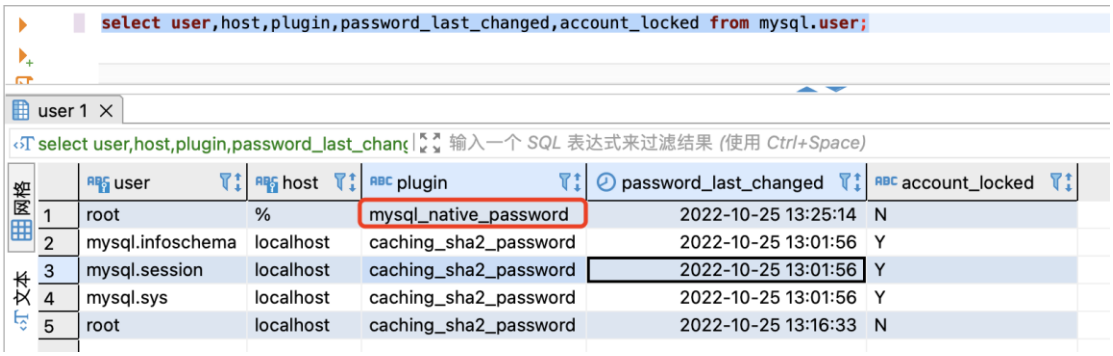
由于 ltloader 暂时不支持 caching_sha2_password 身份认证方式，如果源端 MySQL 身份认证方式为 sha2 需要修改密码认证方式为 mysql_native_password，或者修改 my.cnf 配置文件。MySQL5.7 默认是 mysql_native_password，8.0 开始是 caching_sha2_password。

```
select user,host,plugin,password_last_changed,account_locked from mysql.user;
```




```
#修改加密规则
ALTER USER 'root'@'%' IDENTIFIED BY '123456' PASSWORD EXPIRE NEVER;
#更新密码（mysql_native_password 模式）
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
同时 my.cnf 中加入
default-authentication-plugin=mysql_native_password
create user 'benchmarksql'@'%' identified by '123456';
grant all privileges on *.* to 'benchmarksql'@'%';
```

然后再次执行便可以看到密码验证方式为 mysql_native_password 方式



如果上述方式更改后执行 ltloader 导出仍然失败，报错如下，说明更改认证方式后仍不支持

```
KABOOM!
FATAL error: 76 fell through ECASE expression.
           Wanted one of (2 3 4 5 6 8 9 10 11 14 15 17 20 21 23 27 28 30 31
                          32 33 35 41 42 45 46 47 48 49 50 51 52 54 55 56 60
                          61 62 63 64 65 69 72 77 78 79 82 83 87 90 92 93 94
                          95 96 97 98 101 102 103 104 105 106 107 108 109 110
                          111 112 113 114 115 116 117 118 119 120 121 122 123
                          124 128 129 130 131 132 133 134 135 136 137 138 139
                          140 141 142 143 144 145 146 147 148 149 150 151 159
                          160 161 162 163 164 165 166 167 168 169 170 171 172
                          173 174 175 176 177 178 179 180 181 182 183 192 193
                          194 195 196 197 198 199 200 201 202 203 204 205 206
                          207 208 209 210 211 212 213 214 215 223 224 225 226
                          227 228 229 230 231 232 233 234 235 236 237 238 239
                          240 241 242 243 244 245 246 247 254).
```

则需先将 MySQL8.0 数据迁移至 MySQL5.7

1.7 迁移至 MySQL5.7

首先 MySQL8.0 下使用如下 SQL 进行备份

```
time /root/yc/mysql-8.0.31-linux-glibc2.12-x86_64/bin/mysqldump -uroot -P 8800
-p123456 --databases benchmarksql --column-statistics=0 --set-gtid-purged=OFF
--triggers --master-data=2 --single-transaction --max_allowed_packet=64M >
benchmarksql.sql
```

备份完成后，会在目录中产生 benchmarksql.sql 备份文件，然后还原到 MySQL5.7 中；
首先如果 MySQL5.7 中有 benchmarksql 数据库，需要先手工删除掉，然后再进行创建

```
drop database benchmarksql;
create database benchmarksql;
```

然后执行如下命令进行导入

```
time mysql -uroot -P5757 -p123456 benchmarksql < /root/benchmarksql.sql
```

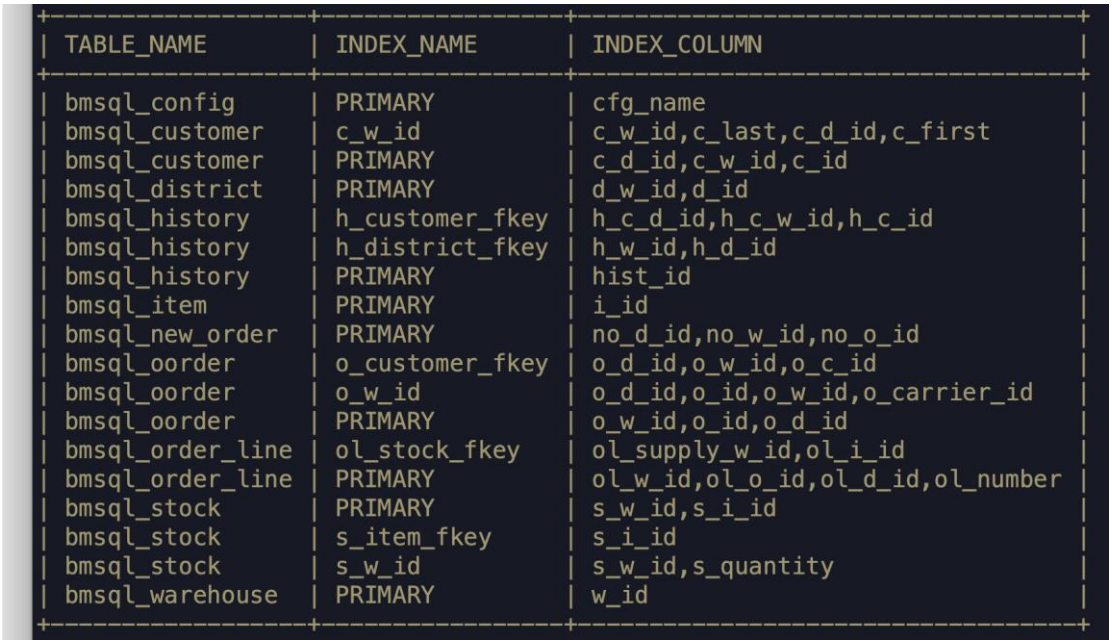
然后使用 MySQL5.7 迁移方式

四、数据校验

1.8 数据校验

统计 MySQL 表和索引数量：

```
SELECT TABLE_NAME, INDEX_NAME, GROUP_CONCAT(COLUMN_NAME) as INDEX_COLUMN
FROM information_schema.statistics
where table_schema='benchmarksql' -- 指定库名
GROUP BY TABLE_NAME, INDEX_NAME;
```



TABLE_NAME	INDEX_NAME	INDEX_COLUMN
bmsql_config	PRIMARY	cfg_name
bmsql_customer	c_w_id	c_w_id,c_last,c_d_id,c_first
bmsql_customer	PRIMARY	c_d_id,c_w_id,c_id
bmsql_district	PRIMARY	d_w_id,d_id
bmsql_history	h_customer_fkey	h_c_d_id,h_c_w_id,h_c_id
bmsql_history	h_district_fkey	h_w_id,h_d_id
bmsql_history	PRIMARY	hist_id
bmsql_item	PRIMARY	i_id
bmsql_new_order	PRIMARY	no_d_id,no_w_id,no_o_id
bmsql_oorder	o_customer_fkey	o_d_id,o_w_id,o_c_id
bmsql_oorder	o_w_id	o_d_id,o_id,o_w_id,o_carrier_id
bmsql_oorder	PRIMARY	o_w_id,o_id,o_d_id
bmsql_order_line	ol_stock_fkey	ol_supply_w_id,ol_i_id
bmsql_order_line	PRIMARY	ol_w_id,ol_o_id,ol_d_id,ol_number
bmsql_stock	PRIMARY	s_w_id,s_i_id
bmsql_stock	s_item_fkey	s_i_id
bmsql_stock	s_w_id	s_w_id,s_quantity
bmsql_warehouse	PRIMARY	w_id

统计 LightDB 表和索引数量：

```
\c benchmarksql benchmarksql
select schemaname,count(*) from pg_tables  where schemaname = 'benchmarksql'
group by schemaname
union all
select schemaname,count(*) from pg_indexes where schemaname = 'benchmarksql'
group by schemaname;

  schemaname  | count
--------------+-------
 benchmarksql |    10
 benchmarksql |    18
```

五、注意点及常见问题

1.9 检测 rum 索引

少数情况下，如 varchar 字段上的 btree 索引在迁移到 LightDB 后会转为 rum 索引，需要手动转换。如下方式可以查找是否有 rum 索引并修改。rum 索引类似于 GIN 索引，主要用于全文检索，在存储上和 ES 更接近，相关度查询时性能比 GIN 索引快 1 倍（注：其缺点是索引大 2 倍，索引速度慢 1/3）。
可通过如下方式修改为其他索引，先查询使用 rum 索引的表，sql 如下：

```
\c benchmarksql benchmarksql
select * from pg_indexes where indexdef like '%rum (%' ;
```

如果执行返回记录如下，则需要更改处理，如果没有则忽略：

```
test=# select * from pg_indexes where indexdef like '%rum (%' ;
 schemaname | tablename |   indexname   | tablespace |
indexdef
------------+-----------+---------------+------------+----------------------
-----------------------------------------
 public     | test_rum  | idx_60507_uid |            | CREATE INDEX idx_60507_uid
ON public.test_rum USING rum (name)
(1 row)
```

复制 indexdef 列的内容，并将 rum 修改为 btree(这里也可以是其他需要的索引类型), 修改后的示例内容如下：

```
CREATE INDEX idx_60507_uid ON public.test_rum USING btree (name);
```

然后删除 rum 索引

```
drop index idx_60507_uid;
```

最后执行保存的 sql，创建对应索引
CREATE INDEX idx_60507_uid ON public.test_rum USING btree (name)

1.10 rum 索引不支持 unique

报错信息如下：

```
2021-11-04T19:47:16.636000+08:00 ERROR PostgreSQL Database error 0A000: access
method "rum" does not support unique indexes
QUERY: CREATE UNIQUE INDEX idx_60455_uid ON test.tab_testtext USING rum(uid);
```

解决方法：复制报错中的 sql，修改类型为 btree，然后在 ltsql 中执行。

```
CREATE UNIQUE INDEX idx_60455_uid ON test.tab_testtext USING btree(uid);
```

1.11 迁移后模式为与数据库同名的模式(schema),能否迁移到其他 schema？

配置文件增加 ALTER SCHEMA，示例如下：
比如对象不放在用户名同名的 benchmarksql schema 下，而是放到 public schema 下

```
loaLOAD DATABASE
FROM mysql://root:123456@10.20.137.41:5757/benchmarksql
INTO postgresql://benchmarksql:benchmarksql@10.20.30.199:5431/benchmarksql
WITH include drop, create tables, create indexes, reset sequences, workers=10,
concurrency=20, max parallel create index =20, foreign keys, prefetch rows =
100000, batch rows = 100000

CAST type int with extra auto_increment to serial drop typemod,
type int to int drop typemod,
type tinyint with extra auto_increment to serial drop typemod,
type smallint with extra auto_increment to serial drop typemod,
type mediumint with extra auto_increment to serial drop typemod,
type bigint with extra auto_increment to bigserial drop typemod,
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 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 numeric,
type double to numeric,
type numeric to numeric keep typemod,
type decimal to decimal keep typemod,
type datetime to timestamp drop default drop not null using zero-dates-to-null
## mysql datetime no timezone
BEFORE LOAD DO
ALTER SCHEMA 'benchmarksql' RENAME TO 'public'
$$ select 1; $$;
```

1.12 Datetime 转换后带时区，如果需要不带时区。

要不带时区，在配置文件里添加如下转换：

```
type datetime to timestamp
```

1.13 数据类型转换列表

蓝色为 MySQL 对应数据类型，

float	double	decimal	tinyint	smallint
numeric	numeric	numeric(10,0)	smallint	smallint
mediumint	int	bigint	int(10)	char(10)
integer	integer	bigint	integer	character(10)
binary(10)	varbinary(10)	tinyblob	mediumblob	longblob
bytea	bytea	bytea	bytea	bytea
text	mediumtext	longtext	enum('female','male')	tinytext
text	text	text	test_u	text
datetime	date	timestamp	year	time
timestamp without time zone	date	timestamp with time zone	integer	time without time zone
json	varchar(10)			
json	character varying(10)			