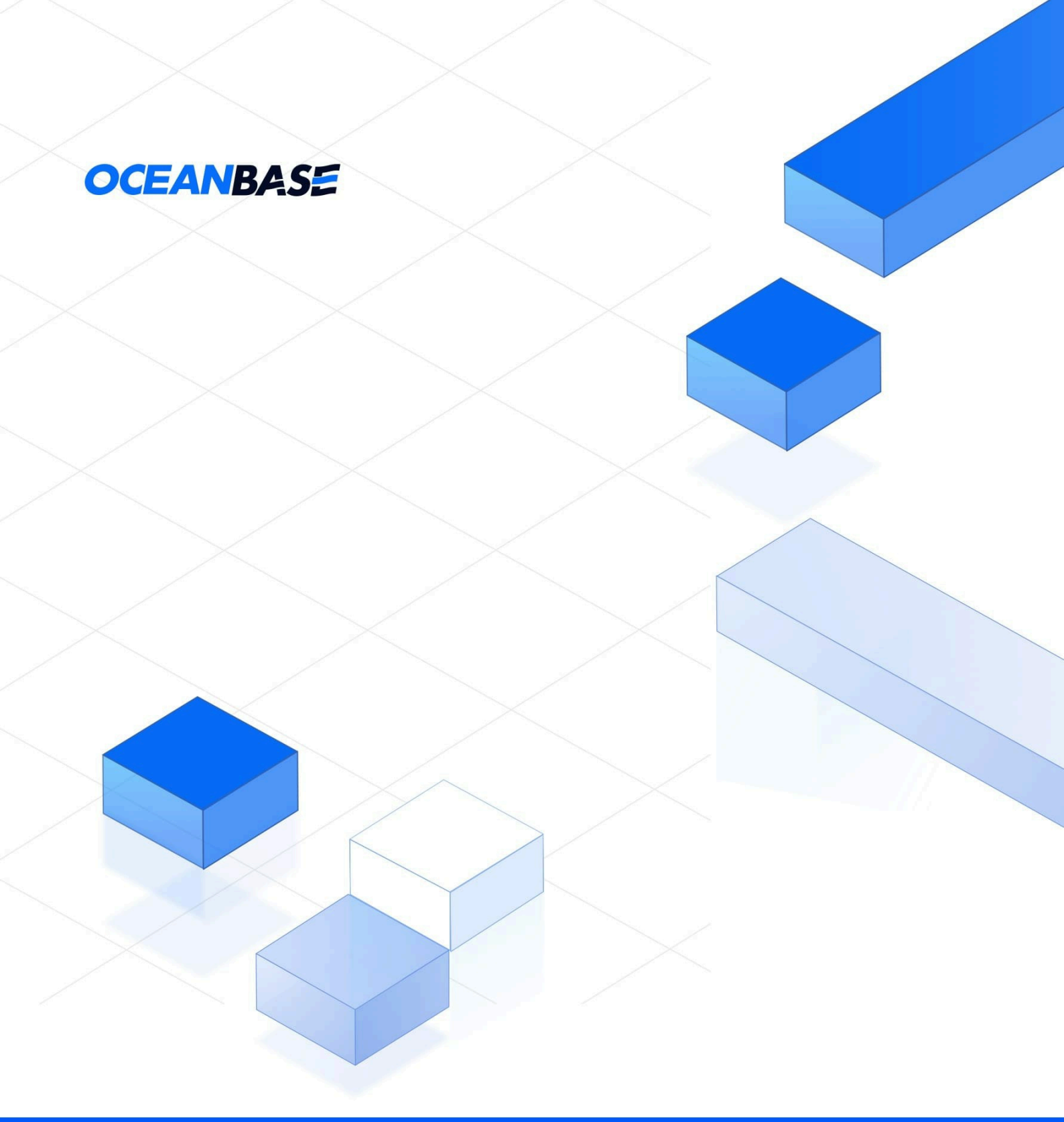


OceanBase 数据库

数据迁移

| 产品版本：V4.0.0 | 文档版本：20230505

声明

蚂蚁集团版权所有©2020，并保留一切权利。

商标声明

免责声明

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	**危险** 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	**警告** 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	**注意** 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	**说明** 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1 关于数据迁移和同步

业务在不断的演进和发展过程中，对数据库的性能和功能要求会发生变化，在更换数据库时，这就会涉及到数据迁移和同步的操作。

数据迁移是日常数据库运维的一种常见操作，是调整集群负载和机房搬迁的必备操作。虽然集群内部、表与表之间数据归档、磁盘水位均衡、资源单元搬迁等操作在 OceanBase 数据库中可以通过简单命令快速发起，但是涉及异构数据源和集群间的数据同步等功能时就需要借助外部工具。

本章将分篇介绍以下几种常用的数据迁移方法及工具：

- 使用 SQL 脚本
  - 常用 SQL 迁移脚本
  - 使用 OUTFILE 语句导出数据
  - 使用 LOAD DATA 语句导入数据
- 使用 mysqldump 迁移
- 使用 DataX 迁移
  - 通用数据同步框架 DataX
  - 不同数据源的 DataX 读写插件示例

2 常用 SQL 迁移脚本

本文介绍如何通过一些常用 SQL 迁移脚本进行数据迁移。

2.1 表与表之间数据迁移

通过 `INSERT INTO……SELECT……` 语句进行数据迁移。

语法格式：

```
INSERT INTO table2
SELECT [(col_name[, col_name] ...)] FROM table1
WHERE [expr];
```

参数	是否必填	描述	示例
INSERT INTO table2	是	数据迁移的目的表	insert into table_B
SELECT [(col_name[, col_name] ...)]	是	选择需要迁移的列，如果要选中全部数据可以用'*'表示。 **注意** 选择的列数量需要与目的表中的列数量保持一致。	select id,name,height
FROM table1	是	数据迁移的源表	from table_A
WHERE [expr]	否	迁移数据的筛选条件；不填表示迁移 select 选中的所有行记录。	where height > 95

语法示例：

下述语句示例展示了如何将表 table_A 中符合条件的数据插入表 table_B 中：

```
obclient> select * from table_A;
+----+------+--------+-------+
| id | name | height | glass |
+----+------+--------+-------+
| 1 | ali | 175 | no |
| 2 | lin | 162 | no |
```

```
| 3 | hei | 172 | no |
+----+------+--------+-------+
3 rows in set

obclient> select * from table_B;
Empty set

obclient> insert into table_B select id,name,height from table_A where height > 170;
Query OK, 2 rows affected
Records: 2 Duplicates: 0 Warnings: 0

obclient> select *from table_B;
+----+------+--------+
| id | name | height |
+----+------+--------+
| 1 | ali | 175 |
| 3 | hei | 172 |
+----+------+--------+
2 rows in set
```

2.2 资源单元迁移

通过 `ALTER SYSTEM` 语句进行资源单元迁移。

- 启动资源单元的迁移

```
obclient> ALTER SYSTEM MIGRATE
UNIT = [unit_id] DESTINATION = [ip_port]
```

- 取消资源单元的迁移

```
obclient> ALTER SYSTEM CANCEL MIGRATE UNIT unit_id;
```

3 使用 OUTFILE 语句导出数据

SELECT INTO OUTFILE 语句能够对需要导出的字段做出限制，满足不需要导出主键字段的场景。配合 LOAD DATA INFILE 语句导入数据，是一种很便利的数据导入导出方式。

3.1 背景信息

OceanBase 数据库 MySQL 模式兼容这一个语法。MySQL 模式下，推荐使用OceanBase 数据库 V2.2.40 及以上版本，可以通过 MySQL Client、OBClient 执行命令。

> **注意**
>
> 客户端需要直连 OceanBase 数据库实例才能执行导出操作。

3.2 语法格式

下述为 `SELECT INTO OUTFILE` 语句的语法：

```
SELECT [column_list] INTO OUTFILE {'/path/file' | 'oss://$PATH/$FILENAME/?
host=$HOST&access_id=$ACCESSID&access_key=$ACCESSKEY'}
[{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[FROM TABLENAME]
[WHERE condition]
[GROUP BY group_expression_list ]
[HAVING condition]]
[ORDER BY order_expression_list]
```

参数	是否必填	描述	示例
SELECT [column_list]	是	选择需要导出的列，如果要选中全部数据可以用'*'表示。	select ID,name,score
'/PATH/FILE' 或 'oss://$PATH/$FILENAME/?host=$HOST&access_id=$ACCESSID&access_key=$ACCESSKEY'	是	选择导出的文件路径，支持导出到阿里云 OSS 中。 **说明** 由于阿里云 OSS 有文件大小的限制，对于超过 5 GB 的文件，导出到 OSS 时会被拆分成多个文件，每个文	into outfile '/home/admin/student.sql'

		件小于 5 GB。	
[{FIELDS | COLUMNS} [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']]	是	指定输出文件中各个字段的格式，通过 Fields | Columns 子句来指定。 ● terminated by：用来指定字段值之间的符号。例如，TERMINATED BY ',' 指定了逗号作为两个字段值之间的标志。 ● enclosed by：用来指定包裹字段值的符号。例如，ENCLOSED BY '"' 表示字符值放在双引号之间。如果使用了 OPTIONALLY 关键字，则仅对字符串类型的值使用指定字符包裹。 ● escaped by：用来指定转义字符。例如，ESCAPED BY '*' 表示将星号（*）指定为转义字符来取代默认的转义字符（\）。	terminated by ',' enclosed by '"'
[LINES [STARTING BY 'string'] [TERMINATED BY 'string']]	是	指定输出文件中每一行的开始和结束字符，通过 Lines 子句设置。 ● starting by：指定每一行开始的字符。 ● terminated by：指定每一行的结束字符。	lines terminated by '\n' 表示一行将以换行符作为结束标志。
[FROM TABLENAME] [WHERE condition] [GROUP BY group_expression_list	是	指定需要导出的表及内容。 ● where：查询的条件 ● group by：分组属性	from student

] [HAVING condition]] [ORDER BY order_expression_list]		• having：分组过滤的条件 • order by：排序属性。	

3.3 使用示例

本文以数据导出到设备本地为例，提供数据的导出示例。

1. 创建表并插入数据。

```
obclient> CREATE TABLE student (ID int primary key,name varchar(128),score int);
Query OK, 0 rows affected

obclient> INSERT INTO student VALUES(1,'lin',98),(2,'hei',90),(3,'ali',95);
Query OK, 3 rows affected
Records: 3 Duplicates: 0 Warnings: 0

obclient> SELECT * FROM student;
+----+------+-------+
| ID | name | score |
+----+------+-------+
| 1 | lin | 98 |
| 2 | hei | 90 |
| 3 | ali | 95 |
+----+------+-------+
3 rows in set
```

2. 使用 root 用户登录到数据库的 sys 租户，设置集群级配置项 secure_file_priv，配置导入或导出文件时可以访问的路径。

   集群级配置项 secure_file_priv 用于控制导入或导出到文件时可以访问的路径。默认值为 NULL，表示导入、导出被禁用。

```
obclient> SET GLOBAL secure_file_priv = "/home/admin";
Query OK, 0 rows affected
```

3. 使用 `SELECT INTO OUTFILE` 语句导出数据。

```
obclient> SELECT ID,name,score INTO OUTFILE '/home/admin/student.sql'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n' FROM student;
Query OK, 3 rows affected
```

4. 登录机器，在设备本地的 `/home/admin` 目录下查看导出的文件信息。

```
[root@***** /home/admin]cat student.sql
1,"lin",98
2,"hei",90
3,"ali",95
```

3.4 更多信息

通过 OUTFILE 方法导出的文件，可以通过 load data 进行导入，详细方法，请参考 使用 LOAD DATA 导入数据。

4 使用 LOAD DATA 语句导入数据

OceanBase 数据库支持通过 LOAD DATA 命令加载外部文件的数据到数据库表中。

4.1 使用场景

LOAD DATA 目前可以对 CSV 格式的文本文件进行导入，整个导入的过程如下：

> **注意**
>
> CSV 文件必须在位于 OBServer 本地，当前版本不支持从远程客户端加载数据。

1. 解析文件。

   OceanBase 会根据用户输入的文件名，读取文件中的数据，并且根据用户输入的并行度来决定并行或者串行解析输入文件中的数据。

2. 分发数据。

   由于 OceanBase 是分布式数据库系统，各个分区的数据可能分布在各个不同的 OBServer 上，LOAD DATA 会对解析出来的数据进行计算，决定数据需要被发送到哪个 OBServer。

3. 插入数据。

   当目标 OBServer 收到了发送过来的数据之后，在本地执行 INSERT 操作把数据插入到对应的分区当中。

4.2 LOAD DATA

4.2.1 获取 LOAD DATA 执行权限

用户需要授予权限才能访问机器上文件，操作步骤如下：

1. 使用 `root` 用户登录到数据库的 `sys` 租户，设置集群级配置项 `secure_file_priv`，配置导入文件时可以访问的路径。

   集群级配置项 `secure_file_priv` 用于控制导入或导出到文件时可以访问的路径。默认值为 `NULL`，表示导入、导出被禁用。

   > **注意**
   >
   > 由于安全原因，以上 SQL 只能使用本地访问执行，不能使用远程 obclient 执行。即需要在 observer 所在机器上登录 obclient （或者 MySQL 客户端） 执行。

```
SET GLOBAL SECURE_FILE_PRIV = "/home/admin";
```

2. 对用户进行授权。

   授予用户 file 权限命令如下，`user_name` 是需要执行 LOAD DATA 命令的用户。

   ```
   GRANT FILE ON *.* TO user_name;
   ```

3. 对待导入文件所在的目录进行授权。

   假设导入文件所在的目录为 `/home/admin/test`，示例如下：

   ```
   chmod -R 755 /home/admin/test
   ```

4.2.2 LOAD DATA 语法格式

```
LOAD DATA
[/*+ parallel(N)*/]
INFILE 'file_name'
[REPLACE | IGNORE] //REPLACE、IGNORE 选项只适用于 MySQL 租户。
INTO TABLE tbl_name
[{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']+
]
[IGNORE number {LINES | ROWS}]
[(col_name_var
[, col_name_var] ...)]
```

参数	是否必填	说明	示例

[/*+ parallel(N)*/]	否	指定加载数据的并行度，建议使用的值范围是[0 - 租户的最大CPU数]	/*+ parallel(4) */
INFILE 'file_name'	是	指定输入文件的路径和文件名。	infile '/home/admin/a.csv'
[REPLACE | IGNORE]	否	指定如何处理重复的数据。LOAD DATA 通过表的主键来判断数据是否重复，如果表不存在主键，那么 LOAD DATA 语句就无法判断数据是否重复，`replace` 和 `ignore`选项没有区别。 ● replace：表示将表中原有的数据替换成为输入文件中的数据。 ● ignore：表示忽略掉重复的数据。 **说明** 如果用户不指定这个选项，那么遇到重复数据的时候，LOAD DATA 语句会将出现把错误的数据记录到日志文件中。	replace
INTO TABLE tbl_name	是	指定目标表名称。	into table tbl_name

[{FIELDS | COLUMNS} [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']]	否	指定输入文件中各个字段的格式，通过 Fields | Columns 子句来指定。 ● terminated By：用来指定字段值之间的符号。例如，TERMINATED BY ',' 指定了逗号作为两个字段值之间的标志。 ● enclosed By：用来指定包裹字段值的符号。例如，ENCLOSED BY ' " ' 表示字符值放在双引号之间。如果使用了 OPTIONALLY 关键字，则仅对字符串类型的值使用指定字符包裹。 ● escaped By：用来指定转义字符。 **说明** 如果用户没有指定，默认值是 terminated by '\t' enclosed by '' escaped by '\\'	fields terminated by ','

[LINES [STARTING BY 'string'] [TERMINATED BY 'string']]	否	指定输入文件中每一行的开始和结束字符，通过 Lines 子句设置。 ● starting By：指定每一行开始的字符。 ● terminated By：指定每一行的结束字符。 **说明** 如果没有指定，默认执行 starting by '' terminated by '\n'	lines terminated by '\n'
[IGNORE number {LINES | ROWS}]	否	子句指定忽略掉输入文件的前 number 行数据。	ignore 1 lines
[(col_name_var [, col_name_var] ...)]	否	指定目标表的各列与输入文件字段之间的关系，通过 (col_name_var [, col_name_var] ...) 关键字指定。 如果用户没有指定，默认会将输入文件中的字段逐个与表中的列进行对应。如果用户通过 col_name_or_user_var 指定输入文件中的字段与表中列的对应关系，LOAD DATA 会根据指定的列名与表中的列进行对应，没有被指定的列会取空值。 如果输入文件中并没有包含所有的列，那么缺少的列按照以下的规则会被默认填充： 字符类型：空字符串 数值类型：0 日期类型：0000-00-00 如果用户需要添加空值，请在输入文件中使用'\N'。	(id, names)

4.2.3 日志文件

如果导入的过程中出现了错误，出现错误的 INSERT 语句会被回滚，并且 LOAD DATA 语句会在 OBServer 安装路径的 log 子目录下产生名称为 obloaddata.log.<XXXXXX> 的日志文件。以下是一个日志文件的内容示例，日志中会包含 LOAD DATA 产生的任务的基本信息，包含租户名、输入文件名、目标表名、并行度、使用的 LOAD DATA 命令，并且以行为单位给出具体错误的信息。

```
Tenant name: mysql
File name: /home/admin/a.csv
Into table: `test`.`t`
Parallel: 1
Batch size: 1000
SQL trace: YD7A20BA65670-0005AADAAA3C****
Start time: 2020-07-29 21:08:13.073741
Load query:
load data infile '/home/admin/test.csv' into table t fields terminated by ',' lines
terminated by '\n'
Row ErrCode ErrMsg
1 1062 Duplicated primary key
2 1062 Duplicated primary key
```

4.3 使用示例

```
[root@***** /home/admin]
$cat student.sql
1,"lin",98
2,"hei",90
3,"ali",95

[root@***** /home/admin]
$obclient -h10.0.0.0 -uroot@MySQL -P2881 -p******
```

```
obclient> create table student_1 (ID int primary key,name varchar(128),score int);

obclient> LOAD DATA INFILE '/home/admin/student.sql' INTO TABLE student_1
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
(ID,name,score);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Deleted: 0 Skipped: 0 Warnings: 0

obclient> select * from student_1
+----+------+-------+
| ID | name | score |
+----+------+-------+
| 1 | lin | 98 |
| 2 | hei | 90 |
| 3 | ali | 95 |
+----+------+-------+
3 rows in set (0.00 sec)
```

5 使用 mysqldump 迁移

mysqldump 是 MySQL 最常用的逻辑导入导出的工具，主要用于转储数据库。

5.1 mysqldump

mysqldump 备份原理是通过协议连接到数据库后，将需要备份的数据查询出来，并将查询的数据转换成对应的 `INSERT` 语句（逻辑导出）。

还原数据时，执行导出文件的 SQL 语句即可 （逻辑导入）。通过此方法，可在不同租户之间，不同数据库之间进行数据迁移。

OceanBase 数据库兼容 MySQL 协议，您可以使用 mysqldump 对 OceanBase 数据库中的数据进行备份。

> **注意**
>
> 使用 mysqldump 仅支持导出 OceanBase 数据库 MySQL 模式实例中的数据。

5.1.1 推荐版本

系统	版本
Linux	mysqldump Ver 10.13 Distrib 5.6.37, for Linux (x86_64)
MacOS	mysqldump Ver 10.13 Distrib 5.7.21, for macos10.13 (x86_64)

5.2 数据导出

5.2.2 数据导出语法

```
mysqldump -hxx.xx.xx.xx -P2881 -u'user@tenantname#clustername' -ppassword --skip-triggers --databases db1 db2 db3 --skip-extended-insert > /tmp/data.sql
```

> **说明**
>
> 对于一些默认选项，在参数前使用--skip可以禁用或者取消相对应的效果。

参数	参数缩写	是否必填	描述	示例
--host	-h	是	服务器IP地址	-h192.168.*.*

--port	-P	是	服务器端口号	-P2881
--user	-u	是	MySQL 模式租户的用户名：用户名@租户名#集群名。 **说明** 如果只填写用户名，执行会默认sys 租户。	-uroot@MySQL#cluster1
--password	-p	否	MySQL 密码	-p1qaz@WSX
--databases	无	是	指定数据库。 --all-databases：导出所有数据库，不推荐使用，建议单独指定。 **说明** 实际使用中，可以直接填写数据库名。	--databases db1 db2 db3

--skip-triggers	无	否	禁用触发器。 **说明** OceanBase 数据库目前不支持 trigger 语法，所以如果不指定 `--force` 参数时必须要加上该参数，否则无法导出。	--skip-triggers
--skip-extended-insert	无	否	导出语句为多条 INSERT 格式，否则为 `INSERT INTO table VALUES(...),(...),` 格式。	--skip-extended-insert
>	无	是	指定保存路径	> /tmp/data.sql

另外一些常用的参数：

参数	参数缩写	是否必填	描述	示例
-d	无	否	仅导出表结构不导出数据。	-d
-t	无	否	只导出数据，而不添加CREATE TABLE 语句。	-t
--compact	无	否	压缩模式，产生更少的输出。	--compact
--comments	无	否	添加注释信息。	--comments
--complete-insert	无	否	输出完成的插入语句。	--complete-insert
--force	无	否	忽略错误。	--force
--lock-tables	无	否	在导出过程中锁库。 --skip-lock-tables：在导出过程中不锁库。 **说明 当前版本OceanBase 数据库中，无论使用哪个参数，在导出过程中都不会锁库。**	--lock-tables

5.3 数据导入

登录数据库，执行以下命令进行导入：

```
source /tmp/data.sql
```

5.4 示例

5.4.3 OceanBase 迁移示例

- 使用 mysqldump 迁移 MySQL 表到 OceanBase

  导出指定数据库的表结构（不包括数据）

  ```
  mysqldump -h 127.1 -uroot -P2881 -p****** --skip-triggers -d TPCH --compact > tpch_ddl.sql
  ```

  在导出的脚本中：

  - 视图的定义也会在里面，但是会以注释（ `/ ! /` ）形式被标注。您无须关注视图部分，可将该部分内容删除。
  - 有一些语法 OceanBase MySQL 不支持，但不影响，替换掉其中部分即可。

  示例如下：

  ```
  /*!40101 SET character_set_client = @saved_cs_client */;
  /*!40101 SET @saved_cs_client = @@character_set_client */;
  /*!40101 SET character_set_client = utf8 */;
  CREATE TABLE `NATION` (
  `N_NATIONKEY` int(11) NOT NULL,
  `N_NAME` char(25) COLLATE utf8_unicode_ci NOT NULL,
  `N_REGIONKEY` int(11) NOT NULL,
  `N_COMMENT` varchar(152) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`N_NATIONKEY`)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
  MAX_ROWS=4294967295;
  ```

  在示例中，导出的脚本里有一个 `MAX_ROWS=` 的设置，这个是 MySQL 特有的，OceanBase MySQL 没有这个问题，也不需要这个设置，且不支持这个语法。您需要把所有 `MAX_ROWS=` 以及后面部分注释掉。注释方式可以使用批量替换技术，例如，在 Vim 中使用 `:%s/MAX_ROWS=/; -- MAX_ROWS=/g` 。

注意

上面导出的 SQL 中表名是大写，说明源端 MySQL 设置表名默认很可能是大小写敏感，因此目标 OceanBase MySQL 租户也要进行同样的设置。

在导出的表结构语句里，可能包含外键。在导入 OceanBase MySQL 里时，如果外键依赖的表没有创建，导入脚本会报错，因此在导入之前需要禁用外键检查约束。

```
MySQL [oceanbase]> set global foreign_key_checks=off;
Query OK, 0 rows affected (0.01 sec)

MySQL [oceanbase]> show global variables like '%foreign%';
+--------------------+-------+
| Variable_name | Value |
+--------------------+-------+
| foreign_key_checks | OFF |
+--------------------+-------+
1 row in set (0.00 sec)
```

在 obclient 里通过 `source` 命令可以执行外部 SQL 脚本文件。

- 导出指定数据库的表数据（不包括结构）

```
mysqldump -h 127.1 -uroot -P2881 -p****** -t TPCH > tpch_data.sql
```

mysqldump 导出的数据初始化 SQL 里会首先将表锁住，禁止其他会话写。然后使用 `insert` 写入数据。每个 `insert` 后面的 `value` 里会有很多值。这是批量 `insert` 。

注意

当前版本 OceanBase 数据库支持 LOCK TABLES 语法输入，但是实际并未对数据库进行锁表，其他会话仍然可以进行写入。

```
LOCK TABLES `t1` WRITE;
/*!40000 ALTER TABLE `t1` DISABLE KEYS */;
INSERT INTO `t1` VALUES ('a'),('中');
/*!40000 ALTER TABLE `t1` ENABLE KEYS */;
```

```
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

5.4.4 常用示例

- 备份所有数据库：

```
mysqldump -uroot -p --all-databases > /backup/mysqldump/alldb.sql
```

- 备份指定数据库 test：

```
mysqldump -uroot -p test > /backup/mysqldump/test.sql
```

- 备份指定数据库指定表 table1，table2。(多个表以空格间隔)

```
mysqldump -uroot -p mysql table1 table2 > /backup/mysqldump/2table.sql
```

- 备份指定数据库排除某些表

```
mysqldump -uroot -p test --ignore-table=test.t1 --ignore-table=test.t2 > /backup
/mysqldump/test2.sql
```

5.5 常见问题

数据处理超时

当您需要导出的数据量较多时，可能会报 `TIMEOUT 4012` 错误。为避免这个错误您需要使用租户管理员账户登录数据库运行下述语句调整系统参数，导出完成后您可以修改回原值：

```
obclient> SET GLOBAL ob_trx_timeout=1000000000,GLOBAL
ob_query_timeout=1000000000;
```

6 通用数据同步框架 DataX

DataX 是阿里巴巴集团内部广泛使用的离线数据同步工具/平台，它支持 MySQL、Oracle、HDFS、Hive、OceanBase、HBase、OTS、ODPS 等各种异构数据源之间高效的数据同步功能。

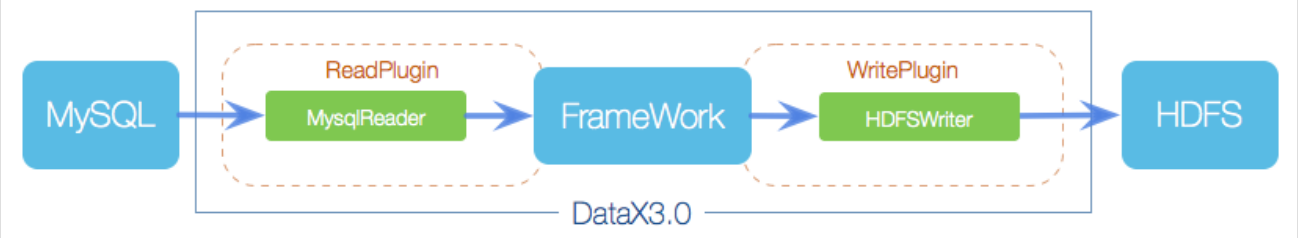
6.1 DataX 简介

DataX 本身作为数据同步框架，将不同数据源的同步抽象为从源头数据源读取数据的 Reader 插件，以及向目标端写入数据的 Writer 插件，理论上 DataX 框架可以支持任意数据源类型的数据同步工作。同时 DataX 插件体系作为一套生态系统，新接入的数据源即可实现和现有的数据源互通。

基于 DataX 的同步机制，可以通过 OceanBase 数据库的 Reader 和 Writer 插件实现 OceanBase 数据库跨数据库、集群和异构数据库的数据迁移。

DataX 安装使用，请参考 DataX-userGuid。DataX 已经在 github 开源，更多详细内容请参考 Alibaba-datax。DataX 还有其商业版 Dataworks（大数据开发治理平台），提供了专业高效、安全可靠的一站式大数据开发与治理平台。

6.2 DataX 框架设计



DataX 作为离线数据同步框架，采用 Framework + Plugin 模式构建。将数据源读取和写入抽象为 Reader/Writer 插件，纳入到整个同步框架中。

- Reader 作为数据采集模块，负责采集数据源的数据，将数据发送给 Framework。
- Writer 作为数据写入模块，负责不断向 Framework 获取数据，并将数据写入到目的端。
- Framework 用于连接 Reader 和 Writer，作为两者的数据传输通道，并处理缓冲、流控、并发、数据转换等核心技术问题。

6.3 DataX 使用示例

DataX 安装后，默认目录在 /home/admin/datax3 。该目录下有个子目录 job ，其中的 job.json 配置文件可以用来验证 DataX 是否安装成功。

每个任务的参数存放在一个 json 格式的文件中，主要由一个 reader 和一个 writer 组成。以下是任务配置示例文件 job.json：

```
[admin /home/admin/datax3/job]
$cat job.json
{
"job": {
"setting": {
"speed": {
"byte":10485760
},
"errorLimit": {
"record": 0,
"percentage": 0.02
}
},
"content": [
{
"reader": {
"name": "streamreader",
"parameter": {
"column" : [
{
"value": "DataX",
"type": "string"
},
{
"value": 19700604,
"type": "long"
},
{
"value": "1970-06-04 00:00:00",
```

```
"type": "date"
},
{
"value": true,
"type": "bool"
},
{
"value": "test",
"type": "bytes"
}
],
"sliceRecordCount": 100000
}
},
"writer": {
"name": "streamwriter",
"parameter": {
"print": false,
"encoding": "UTF-8"
}
}
}
]
}
}
```

这个任务的配置文件的 reader 和 writer 类型是一个 stream，会检测 DataX 是否正确安装。

注意

运行之前请确保安装 JDK 1.8 和 Python 2.7 版本的运行环境。

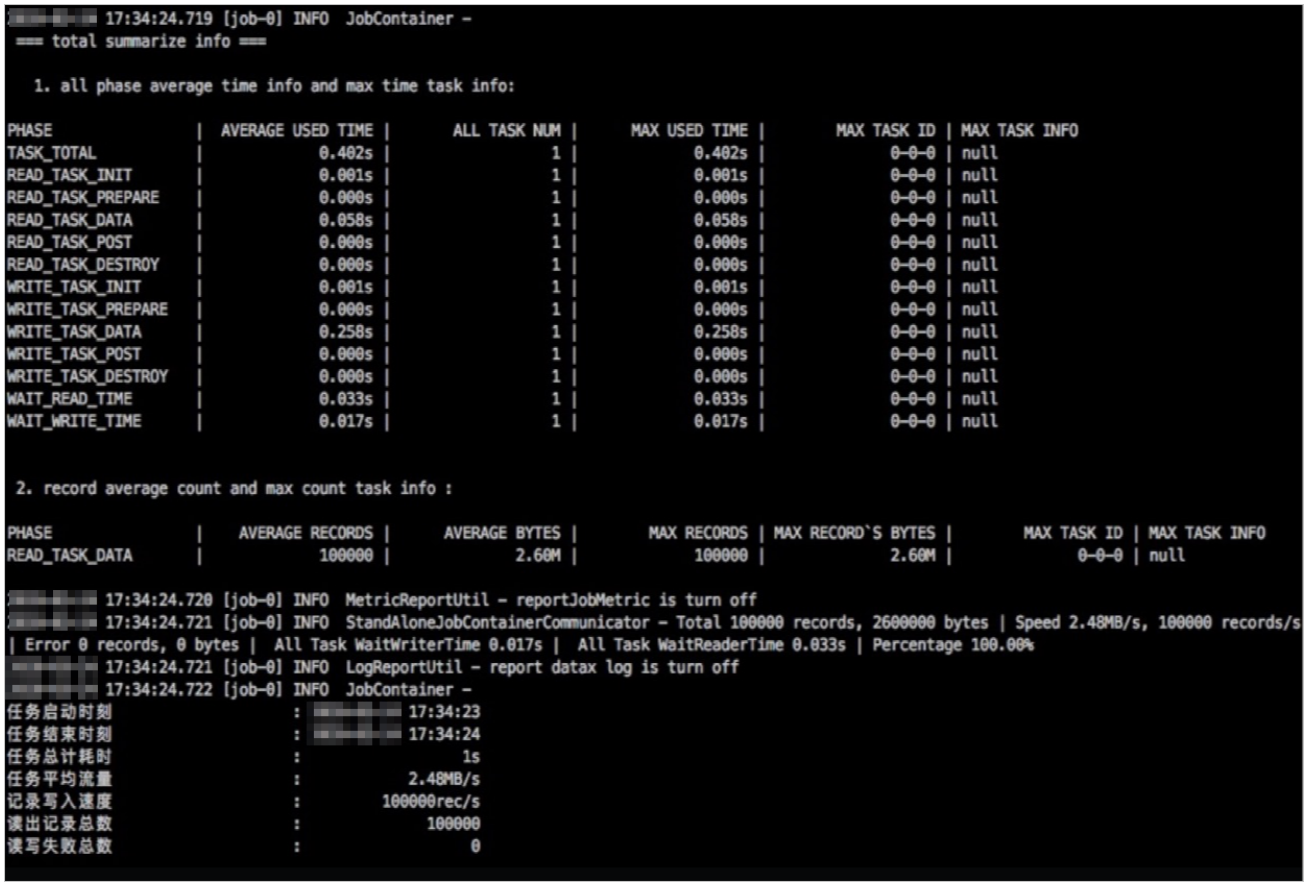
```
```

```
[admin@**** /home/admin/datax3]
$bin/datax.py job/job.json
```

运行结果如下：

说明

这个默认的任务参数关闭了数据流输出，所以看不到输出结果。



7 OceanBase 数据库 DataX 使用示例

OceanBase 数据库使用插件 oceanbasev10reader 和 oceanbasev10writer 来读写。

7.1 oceanbasev10reader 配置示例

```
"reader":{
"name":"oceanbasev10reader",
"parameter":{
"where":"",
"timeout":10000,
"readBatchSize":100000,
"readByPartition":"true",
"column": [
"列名1","列名2"
],
"connection":[
{
"jdbcUrl":[
"||_dsc_ob10_dsc_||集群名:租户名||_dsc_ob10_dsc_||jdbc:oceanbase://连接IP:连接端口/模式名或数据库名"
],
"table":[
"表名"
]
}
],
"username":"租户内用户名",
"password":"******"
```

```
}
}
```

示例：

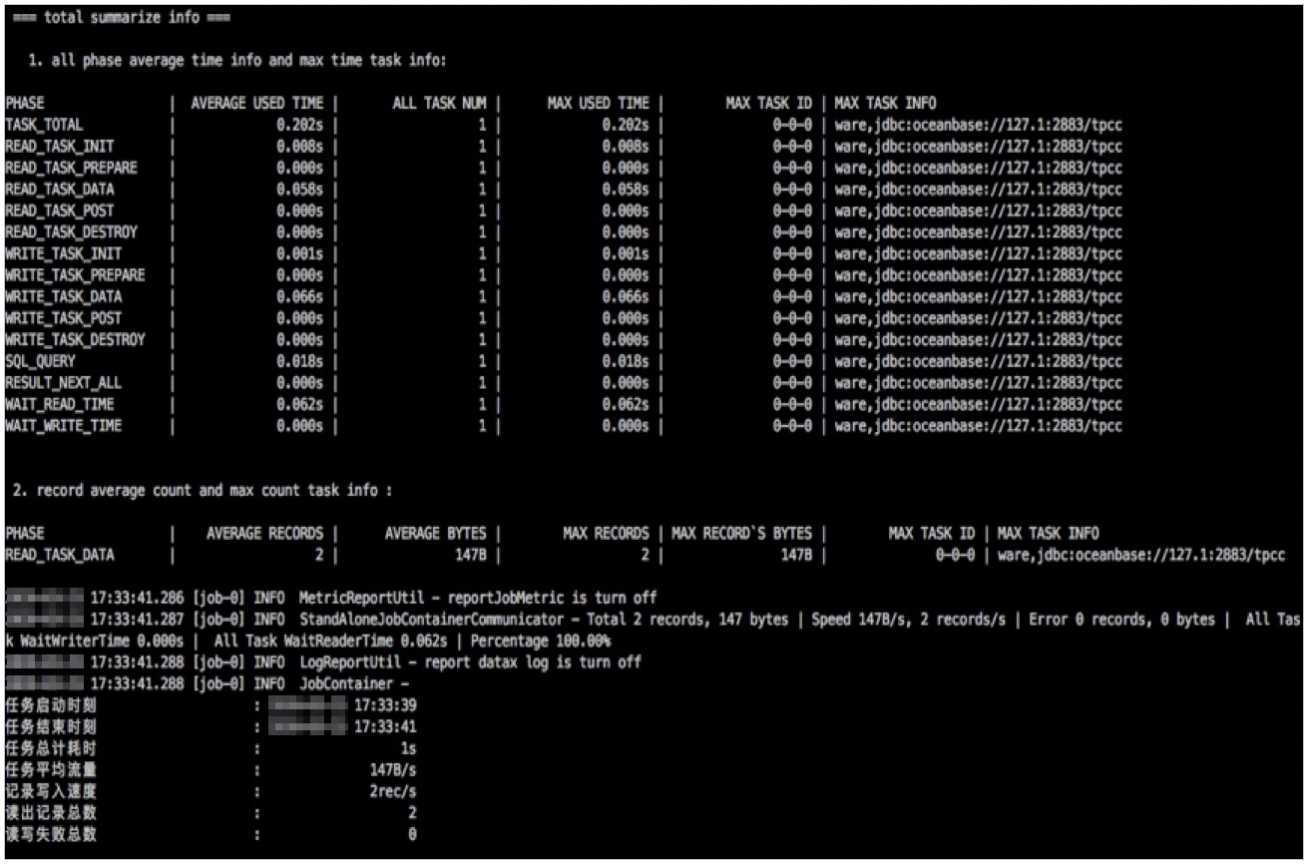
从 OceanBase 数据库中将表 ware 导出到 CSV 文件。

```
[admin@*** /home/admin/datax3]
$cat job/ob_tpcc_ware_2_csv.json
{
"job":{
"setting":{
"speed":{
"channel":10
},
"errorLimit":{
"record":0, "percentage": 0.02
}
},
"content":[
{
"reader":{
"name":"oceanbasev10reader",
"parameter":{
"where":"",
"timeout":10000,
"readBatchSize":100000,
"readByPartition":"true",
"column": [
"W_ID","W_YTD","W_TAX","W_NAME","W_STREET_1","W_STREET_2","W_CITY","
W_STATE","W_ZIP"
],
```

```
"connection":[
{
"jdbcUrl":["||_dsc_ob10_dsc_||obdemo:obbmsql||_dsc_ob10_dsc_||jdbc:
oceanbase://127.1:2883/tpcc"],
"table":["ware"]
}
],
"username":"tpcc",
"password":"******"
}
},
"writer":{
"name":"txtfilewriter",
"parameter":{
"path":"/home/admin/csvdata/",
"fileName":"ware",
"writeMode":"truncate",
"dateFormat":"yyyy-MM-dd",
"charset":"UTF-8",
"nullFormat":"",
"fileDelimiter":"||"
}
}
}
]
}
}
```

```
[admin@*** /home/admin/datax3]
$bin/datax.py job/ob_tpcc_ware_2_csv.json
```

返回结果：



```
=== total summarize info ===

  1. all phase average time info and max time task info:

PHASE              | AVERAGE USED TIME |      ALL TASK NUM |     MAX USED TIME |       MAX TASK ID | MAX TASK INFO
TASK_TOTAL         |             0.202s |                 1 |            0.202s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
READ_TASK_INIT     |             0.008s |                 1 |            0.008s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
READ_TASK_PREPARE  |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
READ_TASK_DATA     |             0.058s |                 1 |            0.058s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
READ_TASK_POST     |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
READ_TASK_DESTROY  |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WRITE_TASK_INIT    |             0.001s |                 1 |            0.001s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WRITE_TASK_PREPARE |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WRITE_TASK_DATA    |             0.066s |                 1 |            0.066s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WRITE_TASK_POST    |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WRITE_TASK_DESTROY |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
SQL_QUERY          |             0.018s |                 1 |            0.018s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
RESULT_NEXT_ALL    |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WAIT_READ_TIME     |             0.062s |                 1 |            0.062s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc
WAIT_WRITE_TIME    |             0.000s |                 1 |            0.000s |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc


 2. record average count and max count task info :

PHASE              |   AVERAGE RECORDS |     AVERAGE BYTES |       MAX RECORDS | MAX RECORD`S BYTES |       MAX TASK ID | MAX TASK INFO
READ_TASK_DATA     |                 2 |              147B |                 2 |              147B |             0-0-0 | ware,jdbc:oceanbase://127.1:2883/tpcc

[redacted] 17:33:41.286 [job-0] INFO  MetricReportUtil - reportJobMetric is turn off
[redacted] 17:33:41.287 [job-0] INFO  StandAloneJobContainerCommunicator - Total 2 records, 147 bytes | Speed 147B/s, 2 records/s | Error 0 records, 0 bytes |  All Tas
k WaitWriterTime 0.000s |  All Task WaitReaderTime 0.062s | Percentage 100.00%
[redacted] 17:33:41.288 [job-0] INFO  LogReportUtil - report datax log is turn off
[redacted] 17:33:41.288 [job-0] INFO  JobContainer -
任务启动时刻                    : [redacted] 17:33:39
任务结束时刻                    : [redacted] 17:33:41
任务总计耗时                    :                1s
任务平均流量                    :             147B/s
记录写入速度                    :            2rec/s
读出记录总数                    :                 2
读写失败总数                    :                 0
```

7.2 oceanbasev10writer 配置示例

使用 DataX 向 OceanBase 数据库里写入时，要避免写入速度过快导致 OceanBase 数据库的增量内存耗尽。

> **说明**
>
> 建议 DataX 配置文件里针对写入做一个写入限速设置，关键字是 memstoreThreshold。

```
"writer": {
"name": "oceanbasev10writer",
"parameter": {
"username": "租户内的用户名",
"password": "******",
"writeMode": "insert",
```

```
"column": [
"列名1","列名2"
],
"preSql": [
""
],
"connection": [
{
"jdbcUrl": "||_dsc_ob10_dsc_||集群名:租户名||_dsc_ob10_dsc_||jdbc:oceanbase://连接IP:连接端口(默认2883)/模式名或数据库名",
"table": [
"表名"
]
}
],
"batchSize": 1024,
"memstoreThreshold":"0.9"
}
}
```

示例：

导入 CSV 文件到 OceanBase 数据库中的 ware 表。

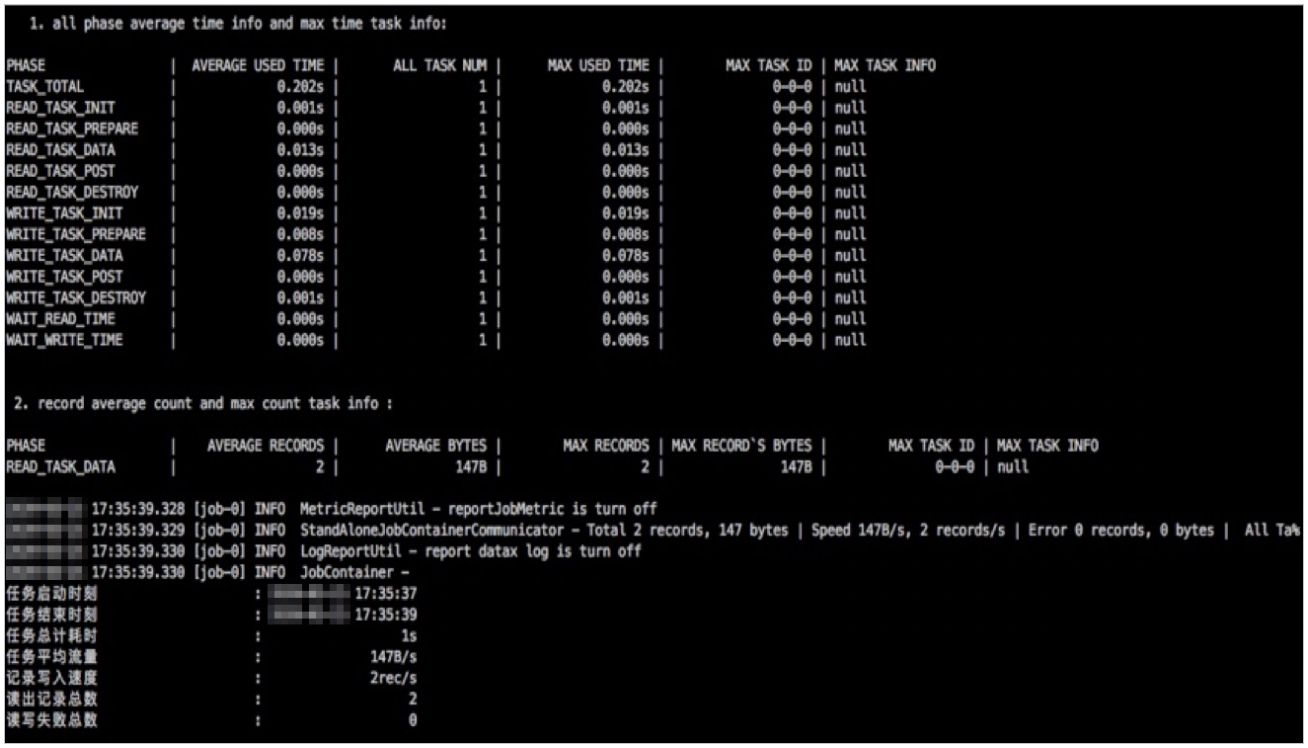
```
[admin@*** /home/admin/datax3]
$cat job/csv_2_ob_tpcc_ware2.json
{
"job":{
"setting":{
"speed":{
"channel":32
},
```

```
"errorLimit":{
"record":0, "percentage": 0.02
}
},
"content":[
{
"reader":{
"name":"txtfilereader",
"parameter":{
"path":["/home/admin/csvdata/ware*"],
"encoding":"UTF-8",
"column":[
{ "index":0, "type":"long" }
,{ "index":1, "type":"long" }
,{ "index":2, "type":"long" }
,{ "index":3, "type":"string" }
,{ "index":4, "type":"string" }
,{ "index":5, "type":"string" }
,{ "index":6, "type":"string" }
,{ "index":7, "type":"string" }
,{ "index":8, "type":"string" }
],
"fieldDelimiter":",",
"fileFormat":"text"
}
},
"writer":{
"name":"oceanbasev10writer",
"parameter":{
```

```
"writeMode":"insert",
"column":[
"W_ID","W_YTD","W_TAX","W_NAME","W_STREET_1","W_STREET_2","W_CITY","
W_STATE","W_ZIP"
],
"connection":[
{
"jdbcUrl":"||_dsc_ob10_dsc_||obdemo:obbmsql||_dsc_ob10_dsc_||jdbc:
oceanbase://127.1:2883/tpcc",
"table":["WARE2"]
}
],
"username":"tpcc",
"password":"******",
"batchSize":256,
"memstoreThreshold":"0.9"
}
}
}
]
}
}


[admin@*** /home/admin/datax3]
$bin/datax.py job/csv_2_ob_tpcc_ware2.json
```

返回结果：



7.3 参数说明

参数	是否必填	描述
jdbcUrl	是	对端数据库的 JDBC 连接信息，使用 JSON 的数组描述，并支持一个库填写多个连接地址。您在 JSON 数组中填写一个 JDBC 连接即可。jdbcUrl 的格式，请参考各数据库官方文档。 **注意** jdbcUrl 必须包含在 connection 配置单元中。
username	是	数据源的用户名。
password	是	数据源指定用户名的密码。
table	是	所选取的需要同步的表。使用 JSON 的数组描述，因此支持多张表同时抽取。当配置为多张表时，用户自己需确保多张表是同一 schema 结构。 **注意** table 必须包含在 connection 配置单元中。

column	是	所配置的表中需要同步的列名集合，使用 JSON 的数组描述字段信息。您可以使用 '*' 代表默认使用所有列配置。 **警告** 不建议将 column 配置为 '*'，在 schema 发生变更时可能会导致任务失败。 支持列裁剪，即列可以挑选部分列进行导出。 支持列换序，即列可以不按照表 schema 信息进行导出。 支持常量配置，用户需要按照Mysql SQL语法格式：["id", "`table`", "1", "'abc.xyz'", "null", "to_char(a + 1)", "2.3" , "true"] 。 ● id：为普通列名。 ● table：为包含保留字的列名。 ● 1：为整形数字常量。 ● abc.xyz：为字符串常量。 ● null：为空指针。 ● to_char(a + 1)：为表达式。 ● 2.3：为浮点数。 ● true：为布尔值。

where	否	筛选条件，MysqlReader 根据指定的 column、table、where 条件拼接 SQL，并根据这个 SQL 进行数据抽取。在实际业务场景中，往往会选择当天的数据进行同步，可以将 where 条件指定为 `gmt_create > $bizdate` 。 **注意** 不可以将 where 条件指定为 limit 10，limit 不是 SQL 的合法 where 子句。where 条件可以有序地进行业务增量同步。如果不填写 where 语句，包括不提供 where 的 key 或者 value，DataX 均视作同步全量数据。

7.4 使用 DataX 迁移 MySQL 数据库到 OceanBase 数据库

将 MySQL 数据迁移到 OceanBase 数据库，如果源端和目标端不能同时跟 DataX 服务器网络联通，那么可以通过 CSV 文件中转。如果源端数据库和目标端数据库能同时跟 DataX 所在服务器联通，则可以使用 DataX 直接将数据从源端迁移到目标端。配置文件如下：

```
{
"job": {
"setting": {
"speed": {
"channel": 4
},
"errorLimit": {
"record": 0,
"percentage": 0.1
}
},
"content": [
{
```

```
"reader": {
"name": "mysqlreader",
"parameter": {
"username": "tpcc",
"password": "********",
"column": [
"列名1","列名2"
],
"connection": [
{
"table": [
"bmsql_oorder"
],
"jdbcUrl": ["jdbc:mysql://127.0.0.1:3306/tpccdb?
useUnicode=true&characterEncoding=utf8"]
}
]
}
},

"writer": {
"name": "oceanbasev10writer",
"parameter": {
"obWriteMode": "insert",
"column": [
"列名1","列名2"
],
"preSql": [
"truncate table bmsql_oorder"
```

```
],
"connection": [
{
"jdbcUrl": "||_dsc_ob10_dsc_||obdemo:oboracle||_dsc_ob10_dsc_||jdbc:
oceanbase://127.0.0.1:2883/tpcc?
useLocalSessionState=true&allowBatch=true&allowMultiQueries=true&rewriteBatched
"table": [
"bmsql_oorder"
]
}
],
"username": "tpcc",
"password":"********",
"writerThreadCount":10,
"batchSize": 1000,
"memstoreThreshold":"0.9"
}
}
}
]
}
}
```

7.5 使用 DataX 迁移 OceanBase 数据库到 MySQL/Oracle 数据库

OceanBase 数据同步到 MySQL

配置文件如下：

```
{
"job": {
"setting": {
"speed": {
"channel": 16
},
"errorLimit": {
"record": 0,
"percentage": 0.1
}
},
"content": [
{
"reader": {
"name": "oceanbasev10reader",
"parameter": {
"where": "",
"readBatchSize": 10000,
"column": [
"列名1","列名2"
],
"connection": [
{
"jdbcUrl": ["||_dsc_ob10_dsc_||obdemo:oboracle||_dsc_ob10_dsc_||jdbc:
oceanbase://127.0.0.1:2883/tpcc"],
"table": [
"bmsql_oorder"
]
}
```

```
],
"username": "tpcc",
"password":"********"
}
},
"writer": {
"name": "mysqlwriter",
"parameter": {
"writeMode": "replace",
"username": "tpcc",
"password": "123456",
"column": [
"列名1","列名2"
],
"session": [
"set session sql_mode='ANSI'"
],
"preSql": [
"truncate table bmsql_oorder"
],
"batchSize": 512,
"connection": [
{
"jdbcUrl": "jdbc:mysql://127.0.0.1:3306/tpccdb?
useUnicode=true&characterEncoding=utf8",
"table": [
"bmsql_oorder"
]
}
```

```
]
}
}
}
]
}
}
```

OceanBase 数据同步到 Oracle

配置文件如下：

```
{
"job": {
"setting": {
"speed": {
"channel": 16
},
"errorLimit": {
"record": 0,
"percentage": 0.1
}
},
"content": [
{
"reader": {
"name": "oceanbasev10reader",
"parameter": {
"where": "",
"readBatchSize": 10000,
"column": [
```

```
"列名1","列名2"
],
"connection": [
{
"jdbcUrl": ["||_dsc_ob10_dsc_||obdemo:oboracle||_dsc_ob10_dsc_||jdbc:
oceanbase://127.0.0.1:2883/tpcc"],
"table": [
"bmsql_oorder"
]
}
],
"username": "tpcc",
"password":"********"
}
},
"writer": {
"name": "oraclewriter",
"parameter": {
"username": "tpcc",
"password": "********",
"column": [
"列名1","列名2"
],
"preSql": [
"truncate table bmsql_oorder"
],
"batchSize": 512,
"connection": [
{
```

```
"jdbcUrl": "jdbc:oracle:thin:@127.0.0.1:1521:helowin",
"table": [
"bmsql_oorder"
]
}
]
}
}
}
]
}
}
```