ALIBABA CLOUD

阿里云

云原生数据仓库AnalyticDB MySQL版

数据接入

文档版本：20201117

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

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.同步数据

AnalyticDB for MySQL提供多种数据同步方案，可满足不同场景下的数据同步需求。

数据源	文档链接
数据库	• 使用DTS同步RDS MySQL数据 • 使用DTS同步DRDS数据 • 使用INSERT外表导出AnalyticDB for MySQL数据到MySQL、同步MySQL数据 • 使用DataWorks同步RDS MySQL数据 • 使用DataWorks同步RDS Server数据 • 使用DataWorks同步Oracle数据
OSS	• 使用INSERT外表同步OSS数据 、通过外表将数据导出到OSS • 使用DataWorks同步OSS数据
大数据	• 使用DataWorks同步Hadoop数据 • 使用DataWorks同步MaxCompute数据 • 使用INSERT外表同步MaxCompute数据
日志数据	• 使用Logstash实时采集日志数据 • 将日志服务数据投递到ADB
消息队列	同步Kafka数据
本地数据	• 使用LOAD DATA导入本地数据 • 使用Kettle将本地数据同步至AnalyticDB for MySQL

2.OSS

2.1. 通过数据集成导入OSS数据

2.1.1. 配置OSS数据源

本文介绍如何在DataWorks中配置OSS数据源。

操作步骤

1. 登录DataWorks控制台，单击目标项目栏中的**进入数据集成**。
2. 在**数据集成**页面，单击左侧导航栏的**数据源**，然后单击**新增数据源**。
3. 在**新增数据源**页面，选择**OSS**。
4. 在**新增OSS数据源**页面，按照页面提示进行参数配置。

参数	说明
数据源名称	为数据源指定一个名字，便于后续管理。
数据源描述	添加数据源描述，该项为可选填项。
Endpoint	OSS EndPoint（地域节点），格式为 `http://oss.aliyuncs.com` 。
Bucket	OSS Bucket名字。
Access Id	OSS Bucket创建者的Access Id。
Access Key	AccessKeyID对应的Key。如何获取AK信息，请参见获取账号的AK信息。

5. 完成上述参数配置后，单击**测试连通性**进行连通性测试，测试通过后单击**完成**。

2.1.2. 配置AnalyticDB for MySQL数据源

您可以参照数据集成同步RDS MySQL数据中的操作步骤完成AnalyticDB for MySQL数据源配置。

详情请参见配置AnalyticDB for MySQL数据源。

2.1.3. 配置同步任务中的数据来源和去向

您可以参照DataWorks同步RDS MySQL中的配置同步任务步骤，将OSS中的数据同步至AnalyticDB for MySQL。

详情请参见使用DataWorks同步RDS MySQL数据。

2.2. 通过外表导入OSS数据

本文介绍如何通过AnalyticDB for MySQL的外部映射表直接查询OSS数据文件，以及如何将OSS中的数据文件导入AnalyticDB for MySQL。目前支持的OSS数据文件格式有Parquet、CSV、TEXT。

前提条件

- 通过以下步骤在对象存储（Object Storage Service，简称OSS）中创建存储AnalyticDB for MySQL数据的目录。
  i. 开通OSS服务

  > **说明** OSS与AnalyticDB for MySQL所属Region相同。

  ii. 创建存储空间
  iii. 新建目录
  iv. 上传测试数据文件

  本示例将 `oss_import_test_data.txt` 文件上传至OSS中的 `bucket-name.oss-cn-hangzhou.aliyuncs.com/adb/` 目录，数据行分隔符为换行符，列分隔符为 `;`，文件示例数据如下所示。

  ```
  number;note
  0001;hello_world_1
  0002;hello_world_2
  0003;hello_world_3
  0004;hello_world_4
  0005;hello_world_5
  0006;hello_world_6
  ...
  ```

- 根据AnalyticDB for MySQL快速入门，完成创建实例、设置白名单、创建账号和数据库等准备工作。

操作步骤

本示例将 `oss_import_test_data.txt` 中的数据导入AnalyticDB for MySQL的 `adb_demo` 库中。

1. 连接目标集群，进入目标数据库。
2. 通过**CREATE TABLE**，在 `adb_demo` 数据库中创建外表。创建CSV、Parquet或TEXT格式OSS外表的建表语法请参见创建OSS外表语法。
3. 查询OSS数据（若仅需导入数据则可跳过此步骤）。

   查询外表映射表和查询AnalyticDB for MySQL内表语法没有区别，您可以方便地直接进行查询，如本步骤的示例代码所示。

   - 对于数据量较大的CSV/TEXT数据文件，强烈建议您按照后续步骤导入AnalyticDB for MySQL后再做查询，否则查询性能可能会较差。
   - 对于Parquet格式数据文件，直接查询的性能一般也比较高，您可以根据需要决定是否进一步导入到AnalyticDB for MySQL后再做查询。

   ```
   select uid, other from oss_parquet_external_table where uid < 100 limit 10
   ```

4. 通过**CREATE TABLE**，在 `adb_demo` 数据库中创建目标表 `adb_oss_import_test` 存储从OSS中导入的数据。

```
CREATE TABLE IF NOT EXISTS adb_oss_import_test
(
  uid string,
  other string
)
DISTRIBUTED BY HASH(uid)
```

5. 执行INSERT语句将OSS数据导入AnalyticDB for MySQL。
   - 执行**INSERT INTO**导入数据。

     ```
     insert into adb_oss_import_test
     select * from oss_import_test_external_table
     ```

   - 执行**INSERT OVERWRITE INTO**导入数据。

     ```
     insert overwrite into adb_oss_import_test
     select * from oss_import_test_external_table
     ```

   - 异步执行**INSERT OVERWRITE INTO**导入数据。

     ```
     submit job insert overwrite into adb_oss_import_test
     select * from oss_import_test_external_table ;
     +---------------------------------------+
     | job_id                                |
     +---------------------------------------+
     | 20201121222029172031009082033030007 15 |
     ```

     关于异步提交任务详情请参见异步提交导入导出任务。

创建OSS外表语法

创建OSS CSV格式外表

示例的 `oss_import_test_data.txt` 文件为CSV格式，本节介绍CSV格式的OSS外表创建语法。

```
CREATE TABLE IF NOT EXISTS oss_import_test_external_table
(
  uid string,
  other string
)
ENGINE='OSS'
TABLE_PROPERTIES='{
  "endpoint":"oss-cn-hangzhou-internal.aliyuncs.com",
  "url":"oss://$bucketname/adb/oss_import_test_data.txt",
  "accessid":"LTAIF****5FsE",
  "accesskey":"Ccw****iWjv",
  "delimiter":";",
  "skip_header_line_count":1
}'
```

参数	说明
ENGINE='OSS'	表示该表是外部表，使用的存储引擎是OSS。
TABLE_PROPERTIES	用于告知AnalyticDB for MySQL如何访问OSS中的数据。
endpoint	OSS的 **EndPoint（域名节点）** 。 **说明** 目前仅支持AnalyticDB for MySQL通过OSS中ECS的VPC网络（内网）访问OSS。 登录OSS控制台，单击目标Bucket，在Bucket**概览**页面查看 **endpoint** 。
url	OSS中目标数据文件或文件夹的绝对地址。建议文件夹地址以 / 结尾。 **示例：** 文件：oss://$Bucket名称/adb/oss_import_test_data.txt 文件夹：oss://$Bucket名称/adb_data/
accessid	您在访问OSS中的文件或文件夹时所持有的AccessKey ID。 如何获取您的accessid和accesskey，请参见获取账号的AK信息。
accesskey	您在访问OSS中的文件或文件夹时所持有的Access Key Secret。
delimiter	定义CSV数据文件的列分隔符。

参数	说明
ossnull（可选）	标识 NULL 值，包含以下四种取值。 • 默认值为1：EMPTY_SEPARATORS a,"",,c --> "a","",NULL,"c" • EMPTY_QUOTES a,"",,c --> "a",NULL,"","c" • BOTH a,"",,c --> "a",NULL,NULL,"c" • NEITHER a,"",,c --> "a","","","c"
skip_header_line_count（可选）	CSV文件第一行为表头，导入数据时设置为1可自动跳过第一行。或设置为其它值表示开头跳过的行数。默认为0，也就是不跳过。

创建OSS Parquet格式外表

```
CREATE TABLE IF NOT EXISTS oss_parquet_external_table
(
  uid string,
  other string
)
ENGINE='OSS'
TABLE_PROPERTIES='{
  "endpoint":"oss-cn-hangzhou-internal.aliyuncs.com",
  "url":"oss://****",
  "accessid":"LTAIF****5FsE",
  "accesskey":"Ccw****iWjv",
  "format":"parquet"
}'
```

参数	说明
ENGINE='OSS'	表示该表是外部表，使用的存储引擎是OSS。
TABLE_PROPERTIES	用于告知AnalyticDB for MySQL如何访问OSS中的数据。

参数	说明
endpoint	OSS的 **EndPoint（域名节点）** 。 **说明** 目前仅支持AnalyticDB for MySQL通过OSS中ECS的VPC网络（内网）访问OSS。 登录OSS控制台，单击目标Bucket，在Bucket**概览**页面查看 **endpoint** 。
url	OSS中目标数据文件或文件夹的绝对地址。建议文件夹地址以 / 结尾。 **示例：** 文件：oss://$Bucket名称/adb/oss_import_test_data.txt 文件夹：oss://$Bucket名称/adb_data/
accessid	您在访问OSS中的文件或文件夹时所持有的AccessKey ID。 如何获取您的accessid和accesskey，请参见获取账号的AK信息。
accesskey	您在访问OSS中的文件或文件夹时所持有的Access Key Secret。
format	数据文件的格式，创建Parquet格式文件的外表时必须将其设置为parquet。

创建Parquet格式文件的外表时，需要注意数据类型的对应关系，具体规则如下：

Parquet基本类型	Parquet的logicalType类型	可对应ADB类型
BOOLEAN	无	boolean
INT32	INT_8	tinyint
INT32	INT_16	smallint
INT32	无	int或integer
INT64	无	bigint
FLOAT	无	float
DOUBLE	无	double
• FIXED_LEN_BYTE_ARRAY • BINARY • INT64 • INT32	DECIMAL	decimal

Parquet基本类型	Parquet的logicalType类型	可对应ADB类型
BINARY	UTF-8	• varchar • string • json（如果已知Parquet该列内容为json格式）
INT32	DATE	date
INT64	TIMESTAMP_MILLIS	timestamp或datetime
INT96	无	timestamp或datetime

注意

- 外表定义中column的名称应与Parquet文件的中该column的名称必须完全对应（可忽略大小写），而顺序可以是随意的，但建议也保持同样顺序。
- 外表定义中的column可以只选择Parquet文件的部分列，未被外表定义的Parquet文件的列将被忽略；反之如果定义了Parquet文件中未包含的列，该列的查询将均为NULL。

创建OSS TEXT格式外表

```
CREATE TABLE IF NOT EXISTS oss_text_external_table
(
  uid string,
  other string
)
ENGINE='OSS'
TABLE_PROPERTIES='{
  "endpoint":"oss-cn-hangzhou-internal.aliyuncs.com",
  "accessid":"LTAIF****5FsE",
  "accesskey":"Ccw****iWjv",
  "format":"text",
  "row_delimiter":"\n",
  "field_delimiter":"\n",
  "URL":"oss://****"
}';
```

参数	说明
ENGINE='OSS'	表示该表是外部表，使用的存储引擎是OSS。
TABLE_PROPERTIES	用于告知AnalyticDB for MySQL如何访问OSS中的数据。

参数	说明
endpoint	OSS的 **EndPoint（域名节点）** 。 **说明** 目前仅支持AnalyticDB for MySQL通过OSS中ECS的VPC网络（内网）访问OSS。 登录OSS控制台，单击目标Bucket，在Bucket**概览**页面查看 **endpoint** 。
url	OSS中目标数据文件的绝对地址。 **示例：** oss://$Bucket名称/adb/oss_import_test_data.txt
accessid	您在访问OSS中的文件时所持有的AccessKey ID。 如何获取您的accessid和accesskey，请参见获取账号的AK信息。
accesskey	您在访问OSS中的文件时所持有的Access Key Secret。
format	数据文件的格式，创建TEXT格式文件的外表时必须将其设置为text。
row_delimiter	定义TEXT文件的行分割符，目前仅支持一种： \n
field_delimiter	定义TEXT文件的列分隔符，只能为一个字符。设置为 \n 表示整行为一个字段。

针对带有分区的Parquet/CSV数据文件创建OSS外表

有的OSS数据源是包含分区的，会在OSS上形成一个分层目录，类似如下内容：

```
parquet_partition_classic/
├── p1=2020-01-01
│   ├── p2=4
│   │   ├── p3=SHANGHAI
│   │   │   ├── 000000_0
│   │   │   └── 000000_1
│   │   └── p3=SHENZHEN
│   │       └── 000000_0
│   └── p2=6
│       └── p3=SHENZHEN
│           └── 000000_0
├── p1=2020-01-02
│   └── p2=8
│       ├── p3=SHANGHAI
│       │   └── 000000_0
│       └── p3=SHENZHEN
│           └── 000000_0
└── p1=2020-01-03
    └── p2=6
        ├── p2=HANGZHOU
        └── p3=SHENZHEN
            └── 000000_0
```

上述数据中p1为第1级分区，p2为第2级分区，p3为第3级分区。对应这种数据源，一般都希望以分区的模式进行查询，那么就需要在创建OSS外表时额外指明分区列。具体的建表语法示例如下（本例为Parquet格式，分区也支持CSV格式）：

```
CREATE TABLE IF NOT EXISTS oss_parquet_partition_table
(
 uid varchar,
 other varchar,
 p1 date,
 p2 int,
 p3 varchar
)
ENGINE='OSS'
TABLE_PROPERTIES='{
 "endpoint":"oss-xxxx.aliyuncs.com",
 "url":"oss://****/****/oss_parquet_data_dir",
 "accessid":"*****",
 "accesskey":"*****",
 "format":"parquet",
 "partition_column":"p1, p2, p3"
}'
```

> **说明**
>
> - 如上例所示，除了在table的列定义中声明p1、p2、p3及其类型，还需要在 TABLE_PROPERTIES 部分中的partition_column属性里声明它们为分区列。且partition_column属性里必须按“第1级, 第2级, 第3级.....”的严格顺序声明（例中p1为第1级分区，p2为第2级分区，p3为第3级分区），在列定义中也需保持相同顺序，并将分区列置于列定义列表的末尾。
> - 可以作为分区列的数据类型有：boolean、tinyint、smallint、int/integer、bigint、float、double、decimal、varchar/string、date、timestamp。
> - 查询时分区列和其它数据列的表现和用法没有区别。

2.3. 通过外表将数据导出到OSS

AnalyticDB for MySQL支持通过外表和INSERT INTO方式将AnalyticDB for MySQL中的数据导出到对象存储OSS（Object Storage Service）中。将数据导出到OSS功能只支持CSV和Parquet格式文件，不支持TEXT格式文件。

前提条件

- 通过以下步骤在对象存储服务OSS中创建存储AnalyticDB for MySQL数据的目录：
  i. 开通对象存储服务，详情请参见开通OSS服务。

  > **说明** OSS与AnalyticDB for MySQL所属Region相同。

  ii. 在OSS中创建存储空间，详情请参见创建存储空间。
  iii. 在OSS中新建目录，详情请参见新建文件夹。

例如，在OSS中新建目录 `adb_data/` ，从AnalyticDB for MySQL中导出的数据将存储在该目录下。

- 根据AnalyticDB for MySQL快速入门，完成创建集群、设置白名单、创建账号和数据库等准备工作，详情请参见AnalyticDB for MySQL快速入门。

操作步骤

本示例将AnalyticDB for MySQL的 `adb_demo` 库中的 `source_table` 表数据导出至OSS的 `adb_data` 文件夹下。

1. 连接目标集群，进入源数据库。
2. 在 `adb_demo` 数据库中创建外表，详情请参见创建OSS外表语法。
3. 根据外表类型选择执行写入语句，将源数据写入到步骤2创建的外表中，不同的外表类型支持的语法请参见未做分区的普通外表语法支持和分区外表语法支持。
4. 待步骤3的写入任务结束后，您可登录OSS控制台，在目标文件夹下查看导出到OSS的数据。

未做分区的普通外表语法支持

说明 本节中各类外表写出功能，对Parquet格式外表写出功能默认关闭。您可通过hint开放写入。

```
/*+oss_parquet_write_enable=true*/ INSERT INTO ...
```

INSERT SELECT FROM

功能：如果您的数据在其他表中已经存在，可以通过 `INSERT SELECT FROM` 将数据复制到外表。

语法：

```
INSERT INTO table_name
SELECT select_statement FROM from_statement;
```

例句：

```
insert into oss_table select col1, col2, col3 from source_table;
```

支持状态	异常信息	功能描述
支持。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	无	将源表的数据写入外表对应的OSS位置，每次写入会产生新的OSS数据文件。

REPLACE SELECT FROM

功能：由于OSS外表不支持定义主键，因此 `REPLACE SELECT FROM` 的写入表现与 `INSERT SELECT FROM` 保持一致，都是将数据复制到另外一张表；如果目标表已有数据，已有数据保持不变，新数据追加到新的OSS数据文件。

语法：

```
REPLACE INTO table_name
SELECT select_statement FROM from_statement;
```

例句：

```
replace into oss_table select col1, col2, col3 from source_table;
```

支持状态	异常信息	功能描述
支持。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	无	将源表的数据写入外表对应的OSS位置，每次写入会产生新的OSS数据文件。

INSERT OVERWRITE INTO SELECT

功能：`INSERT OVERWRITE INTO SELECT` 用于向表中批量插入数据。如果目标外表中已存在数据，则每次写入会先删除原外表路径下全部数据文件，再产生新的OSS数据文件。

语法：

```
INSERT OVERWRITE INTO table_name
SELECT select_statement FROM from_statement;
```

例句：

```
insert overwrite into oss_table
select col1, col2, col3 from source_table;
```

支持状态	异常信息	功能描述
支持，表现为删除根目录下全体文件后再重新写入。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	无	将源表的数据写入外表对应的OSS位置，每次写入会先删除原外表路径下全部数据文件，再产生新的OSS数据文件。

导出到OSS单文件（仅限CSV格式，Parquet格式不允许导出到单一文件）

功能：通过hint指定唯一的OSS文件，将数据导出到此文件中。包含overwrite关键字时，覆盖外表TABLE_PROPERTIES中定义的目录下旧的同名文件，不影响该目录下其他文件。

版本说明：

- 3.1.2版本之前的AnalyticDB for MySQL实例：不支持此功能，文件名由系统自动命名，将会导出多个文件。根据任务并发速度动态确定目标文件数目。
- 3.1.2及之后版本的AnalyticDB for MySQL实例：支持通过hint支持将AnalyticDB for MySQL CSV格式的普通外表导出到OSS单文件，用户在导出时可自定义文件名。不带hint的情况下，与3.1.2之前的版本保持一致（导出多个文件）。

语法：

```
/*output_filename=adb.txt*/INSERT [OVERWRITE] table_name
SELECT select_statement FROM from_statement;
```

例句：

```
/*output_filename=adb.txt*/INSERT [OVERWRITE] oss_table
SELECT * FROM source_table;
```

支持状态	异常信息	功能描述
支持。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	无	将源表的数据写入外表对应的OSS位置，每次写入会产生新的OSS数据文件。

未带hint的各类导出效果：

导出到单一文件的效果：

分区外表语法支持

分区表写出数据时，数据文件内不包含分区列的数据，分区列的数据信息以OSS目录的形式展现。

例如：分区表中定义了2个分区列，3个普通列。其中一级分区名称为pcol1，分区数值为1；二级分区名称为pcol2，分区数值为a。分区表数据导出到OSS的保存路径为*adb_data/*，则向pcol1=1且pcol2=a的外表分区写出数据时，数据文件相对路径目录为：*adb_data/pcol1=1/pcol2=a/*；且外表CSV/Parquet数据文件内不包含pcol1与pcol2这两列的值，只包含3列普通列的值。

> **说明** 本节中各类分区外表写出功能，对CSV和Parquet格式分区外表写出功能均默认关闭。您可通过hint开放写入，CSV和Parquet格式分区外表均使用以下语句：
>
> ```
> /*+oss_parquet_write_enable=true*/ INSERT INTO ...
> ```

INSERT INTO PARTITION SELECT FROM

功能：`INSERT INTO PARTITION INTO SELECT`用于向带分区的外表中批量插入数据。写入时要求指明全部分区信息，且分区的值不能为空。

语法：

```
INSERT into table_name PARTITION(par1=val1,par2=val2,...)
SELECT select_statement FROM from_statement;
```

例句：

```
insert into oss_table_par PARTITION(par1=val1,par2=val2)
select col1, col2, col3 from source_table;
```

支持状态	异常信息	功能描述
支持。写入时，在对应分区追加写入。写入时要求指明全部分区信息，且分区的值不能为空。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	• 当分区信息不全时：Invalid: define partition columns number is partition_num,not equal to insert partition number m。 • 当分区的值为空时：Query execution error: : PARTITION value can not be null。	将源表的数据写入外表对应的OSS位置下的对应分区目录中，每次写入会产生新的OSS数据文件。

REPLACE INTO PARTITION SELECT FROM

功能：由于外表不支持主键，`REPLACE INTO PARTITION SELECT FROM` 行为表现与 `INSERT INTO PARTITION SELECT FROM` 一样。

语法：

```
REPLACE into table_name PARTITION(par1=val1,par2=val2,...)
SELECT select_statement FROM from_statement;
```

例句：

```
REPLACE into oss_table_par PARTITION(par1=val1,par2=val2)
select col1, col2, col3 from source_table;
```

支持状态	异常信息	功能描述
支持。写入时，在对应分区追加写入。写入时要求指明全部分区信息，且分区的值不能为空。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	• 当分区信息不全时：Invalid: define partition columns number is partition_num,not equal to insert partition number m。 • 当分区的值为空时：Query execution error: : PARTITION value can not be null。	将源表的数据写入外表对应的OSS位置下的对应分区目录中，每次写入会产生新的OSS数据文件。

INSERT OVERWRITE [INTO] PARTITION SELECT

功能：`INSERT OVERWRITE [INTO] PARTITION [IF NOT EXISTS] SELECT` 用于向带分区的外表中批量插入数据。写入时要求指明全部分区信息，且分区的值不能为空。如果目标外表中已存在数据，每次写入会先删除原外表对应分区目录下全部数据文件，再产生新的OSS数据文件。

语法：

```
INSERT OVERWRITE [INTO] table_name PARTITION(par1=val1,par2=val2,...)[IF NOT EXISTS]
SELECT select_statement FROM from_statement;
```

例句：

```
INSERT OVERWRITE into oss_table_par PARTITION(par1=val1,par2=val2) IF NOT EXISTS
select col1, col2, col3 from source_table;
```

支持状态	异常信息	功能描述
支持。在对应分区删除旧文件后，再写入。如果带有 `if not exists` ，则先判断分区是否存在，如果存在则不执行。写入时要求指明全部分区信息，且分区的值不能为空。写入的外表必须保持列个数的完整，不允许用户指定只写入一部分的列。	• 当分区信息不全时：Invalid: define partition columns number is partition_num,not equal to insert partition number m。 • 当分区的值为空时：Query execution error: : PARTITION value can not be null。 • 如果带有 `if not exists` ，则先判断分区是否存在，如果存在则报错不执行覆盖写入：partition is already existed。	将源表的数据写入外表对应的OSS位置下的对应分区目录中，每次写入会先删除原外表对应分区目录下全部数据文件，再产生新的OSS数据文件。如果带有 `if not exists` ，则先判断分区是否存在，如果存在则不执行。

不支持语法

ADB for MySQL 3.0 不支持自定义行级写入的插入语法，具体不支持的语法如下：

INSERT INTO VALUES

语法：

```
INSERT [IGNORE]
  INTO table_name
  [( column_name [, ···] )]
  [VALUES]
  [(value_list[, ···])]
  [query];
```

REPLACE INTO VALUES

语法：

```
REPLACE
  INTO table_name
  [(column_name,...)]
  VALUES
  ({常量|NULL|DEFAULT},...),(...),...
```

分区外表暂不支持未指定分区的写入INSERT SELECT FROM

语法：

```
INSERT INTO table_name
SELECT select_statement FROM from_statement;
```

分区外表不支持导出到OSS单文件

语法：

```
/*output_filename=adb.txt*/INSERT into table_name PARTITION(par1=val1,par2=val2,...)
SELECT select_statement FROM from_statement;
/*output_filename=adb.txt*/REPLACE into table_name PARTITION(par1=val1,par2=val2,...)
SELECT select_statement FROM from_statement;
/*output_filename=adb.txt*/INSERT OVERWRITE [INTO] table_name PARTITION(par1=val1,par2=val2,...)
[IF NOT EXISTS]
SELECT select_statement FROM from_statement;
```

3.大数据

3.1. 通过DataWorks导入EMR数据

3.1.1. 配置AnalyticDB for MySQL数据源

您可以参照DataWorks同步RDS MySQL数据中的操作步骤完成AnalyticDB for MySQL数据源配置。

详情请参见配置AnalyticDB for MySQL数据源。

3.1.2. 配置同步任务中的数据来源和去向

请参见DataWorks同步RDS MySQL中的配置同步任务步骤，将EMR中的数据同步至AnalyticDB for MySQL。

详情请参见使用DataWorks同步RDS MySQL数据。

3.1.3. 配置HDFS数据源

本文介绍如何在DataWorks中配置HDFS数据源。

操作步骤

1. 登录DataWorks控制台，单击目标项目栏中的**进入数据集成**。
2. 在**数据集成**页面，单击左侧导航栏的**数据源**，然后单击**新增数据源**。
3. 在**新增数据源**页面，选择**HDFS**。
4. 在**新增HDFS数据源**页面，按照页面提示进行参数配置。

参数	说明
数据源名称	数据源名称必须包含字母、数字、下划线，但不能以数字和下划线开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
DefaultFS	nameNode节点地址，格式为 `hdfs://ServerIP:Port` 。

5. 单击**测试连通性**。
6. 测试连通性通过后，单击**完成**。

   测试连通性说明

   - 经典网络ECS上自建的数据源，建议使用数据集成自定义资源组，默认资源组不保证网络可通。
   - 专有网络目前不支持数据源连通性测试，直接单击**完成**。

3.2. 通过DataWorks导入Hadoop数据

3.2.1. 配置HDFS数据源

本文介绍如何在DataWorks中配置HDFS数据源。

操作步骤

1. 登录DataWorks控制台，单击目标项目栏中的**进入数据集成**。
2. 在**数据集成**页面，单击左侧导航栏的**数据源**，然后单击**新增数据源**。
3. 在**新增数据源**页面，选择**HDFS**。
4. 在**新增HDFS数据源**页面，按照页面提示进行参数配置。

参数	说明
数据源名称	数据源名称必须包含字母、数字、下划线，但不能以数字和下划线开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
DefaultFS	nameNode节点地址，格式为 `hdfs://ServerIP:Port` 。

5. 单击**测试连通性**。
6. 测试连通性通过后，单击**完成**。

   测试连通性说明

   - 经典网络ECS上自建的数据源，建议使用数据集成自定义资源组，默认资源组不保证网络可通。
   - 专有网络目前不支持数据源连通性测试，直接单击**完成**。

3.2.2. 配置AnalyticDB for MySQL数据源

您可以参照DataWorks同步RDS MySQL数据中的操作步骤完成AnalyticDB for MySQL数据源配置。

详情请参见配置AnalyticDB for MySQL数据源。

3.2.3. 配置同步任务中的数据来源和去向

请参见DataWorks同步RDS MySQL中的配置同步任务步骤，将Hadoop中的数据同步至AnalyticDB for MySQL。

详情请参见使用DataWorks同步RDS MySQL数据。

3.3. AnalyticDB for MySQL与MaxCompute数据导入导出

3.3.1. 通过INSERT外表方式导出到MaxCompute分区表

本文介绍如何通过INSERT外表方式将AnalyticDB for MySQL数据导出到MaxCompute（原名ODPS）分区表。

前提条件

- 根据MaxCompute准备工作和快速入门准备目标数据表。

例如通过创建表语句，在MaxCompute中创建以下表。如果您已经有目标数据表，请跳过该步骤。

- 一级分区表

```
CREATE TABLE IF NOT EXISTS odps_table
(
uid STRING,
other STRING,
ds STRING
)
PARTITIONED BY (ds STRING)
LIFECYCLE 3;
```

- 二级分区表

```
CREATE TABLE IF NOT EXISTS odps_table
(
uid STRING,
other STRING,
ds STRING
)
PARTITIONED BY (ds STRING,other STRING)
LIFECYCLE 3;
```

- 根据AnalyticDB for MySQL快速入门，完成创建实例、设置白名单、创建账号和数据库等准备工作。

> **说明** 导出数据到MaxCompute分区表时，需要明确指定待写入的分区，不支持同时导出到多个分区，您可以通过执行多个SQL的方式实现将数据导出到多个MaxCompute分区。MaxCompute最高支持6级分区，其他多级分区的操作步骤类似。

导出到MaxCompute一级分区表

本示例将AnalyticDB for MySQL的 `adb_table` 表中的数据导出到MaxCompute的 `odps_table` 一级分区表中。

1. 连接集群，登录数据源AnalyticDB for MySQL数据库。
2. 新建一张外表映射到MaxCompute的目标数据表。

```
CREATE TABLE odps_external_table
(
uid string,
other string,
ds string
)
ENGINE='ODPS'
TABLE_PROPERTIES='{
"endpoint":"http://service.odps.aliyun-inc.com/api",
"accessid":"xxx",
"accesskey":"xxx",
"project_name":"xxx",
"table_name":"odps_table",
"partition_column":"ds"
}'
```

参数	说明
ENGINE='ODPS'	表示该表是外部表，使用的存储引擎是MaxCompute。
TABLE_PROPERTIES	用于告知AnalyticDB for MySQL如何访问MaxCompute，并向其写入数据。
endpoint	MaxCompute的 **EndPoint（域名节点）** **说明** 目前仅支持AnalyticDB for MySQL通过MaxCompute的VPC网络Endpoint访问MaxCompute。 如何查看MaxCompute的EndPoint，请参见配置Endpoint。
accessid	您在访问MaxCompute目标数据表时所持有的AccessKey ID。
accesskey	您在访问MaxCompute目标数据表时所持有的Access Key Secret。
project_name	MaxCompute中目标数据表所在的工作空间名
table_name	MaxCompute中目标数据表的表名
partition_column	分区字段

3. 将数据从AnalyticDB for MySQL外表导出到MaxCompute的数据表一级分区表。

> 说明 adb_table_column 不包含partition里面的字段。

```
insert [overwrite] into odps_external_table partition(ds='20200401')
select [adb_table_column, ...] from adb_table [where ...]
```

导出到MaxCompute二级分区表

本示例将AnalyticDB for MySQL的 adb_table 表中的数据导出到MaxCompute的 odps_table 二级分区表中。

1. 连接集群，进入数据源AnalyticDB for MySQL数据库。
2. 新建一张外表映射到MaxCompute的数据表。

```
CREATE TABLE odps_external_table
(
uid string,
other string,
ds string
)
ENGINE='ODPS'
TABLE_PROPERTIES='{
"endpoint":"http://service.odps.aliyun-inc.com/api",
"accessid":"xxx",
"accesskey":"xxx",
"project_name":"xxx",
"table_name":"odps_table",
"partition_column":"ds,other"
}'
```

3. 将数据从AnalyticDB for MySQL外表导出到MaxCompute数据表二级分区表。

> 说明 adb_table_column 不包含partition里面的字段。

```
insert [overwrite] into odps_external_table partition(ds='20200401',other='hangzhou')
select [adb_table_column, ...] from adb_table [where ...]
```

3.3.2. 通过INSERT外表方式导入MaxCompute数据

本文介绍如何将MaxCompute数据导入AnalyticDB for MySQL中。

前提条件

- 根据MaxCompute准备工作和快速入门准备源数据。

  例如通过创建表语句，在MaxCompute中创建以下表。如果您已经有数据源，请跳过该步骤。

```
CREATE TABLE IF NOT EXISTS odps_nopart_import_test
(
uid STRING,
other STRING
)
LIFECYCLE 3;
```

在DataWorks中，通过数据导入的方式将文件odps_nopart_import_test.txt中的数据导入odps_nopart_import_test表。

- 根据AnalyticDB for MySQL快速入门，完成创建实例、设置白名单、创建账号和数据库等准备工作。

操作步骤

1. 连接AnalyticDB for MySQL集群，进入目标数据库。

   本示例将 `odps_nopart_import_test` 表中的数据导入AnalyticDB for MySQL的 `test_adb` 数据库中。

2. 通过CREATE TABLE，在 `test_adb` 数据库中创建外部映射表 `odps_nopart_import_test_external_table` 。

```
CREATE TABLE IF NOT EXISTS odps_nopart_import_test_external_table
(
 uid string,
 other string
)
 ENGINE='ODPS'
 TABLE_PROPERTIES='{
 "endpoint":"http://service.cn.maxcompute.aliyun-inc.com/api",
 "accessid":"L*******FsE",
 "accesskey":"CcwF********iWjv",
 "project_name":"odps_project1",
 "table_name":"odps_nopart_import_test"
 }'
```

参数	说明
ENGINE='ODPS'	表示该表是外部表，使用的存储引擎是MaxCompute。
TABLE_PROPERTIES	用于告知AnalyticDB for MySQL如何访问MaxCompute中的数据。
endpoint	MaxCompute的**EndPoint（域名节点）**。 **说明** 目前仅支持AnalyticDB for MySQL通过MaxCompute的VPC网络Endpoint访问MaxCompute。 如何查看MaxCompute Endpoint，请参见配置Endpoint。

参数	说明
accessid	您访问MaxCompute源表时所持有的Access Key Secret。 如何获取您的 accessid 和 accesskey ，请参见获取账号的AK信息。
accesskey	您访问MaxCompute源表时所持有的AccessKey ID。
project_name	MaxCompute中的工作空间名称。
table_name	MaxCompute中的数据源表名。

3. 通过CREATE TABLE，在 test_adb 数据库中创建目标表 adb_nopart_import_test 存储从MaxCompute中导入的数据。

```
CREATE TABLE IF NOT EXISTS adb_nopart_import_test
(
uid string,
other string
)
DISTRIBUTE BY HASH(uid);
```

4. 执行INSERT语句将MaxCompute数据导入AnalyticDB for MySQL。
   - 执行 INSERT INTO 导入数据。

```
insert into adb_nopart_import_test
select * from odps_nopart_import_test_external_table
```

```
select * from adb_nopart_import_test
+---------------+-----------------+
| uid       | other       |
+---------------+-----------------+
| 4         | other4      |
| 6         | other6      |
| 5         | other5      |
| 2         | other2      |
| 1         | other1      |
| 3         | other3      |
| 7         | other7      |
```

   - 异步执行 INSERT OVERWRITE INTO 导入数据。

```
submit job insert overwrite into adb_nopart_import_test
select * from odps_nopart_import_test_external_table
+----------------------------------------+
| job_id                                 |
+----------------------------------------+
| 20201121222029172031009082033030000321 |
```

关于异步提交任务详情请参见异步提交导入导出任务。

3.4. 通过DataWorks导入MaxCompute数据

3.4.1. 配置MaxCompute数据源

本文介绍如何在DataWorks中配置MaxCompute数据源。

详情请参见配置MaxCompute数据源。

3.4.2. 配置AnalyticDB for MySQL源

您可以参照数据集成同步RDS MySQL数据中的操作步骤完成AnalyticDB for MySQL数据源配置。

详情请参见配置AnalyticDB for MySQL数据源。

3.4.3. 配置同步任务中的数据来源和去向

您可以参照DataWorks同步RDS MySQL中的配置同步任务步骤，将MaxCompute中的数据同步至AnalyticDB for MySQL。

详情请参见使用DataWorks同步RDS MySQL数据。

3.5. Flink ADB Connector

本文以示例形式介绍如何将开源Flink中的数据写入到ADB。

操作步骤

1. 下载flink-jdbc-adb-compatibility_2.11-1.10.10.jar。
2. 将所有Flink节点 `/home/admin/flink-1.10.1/lib` 下的 `flink-jdbc_2.11-1.10.1.jar` 文件替换为flink-jdbc-adb-compatibility_2.11-1.10.10.jar 。
3. 重启所有Flink节点。
4. 在Flink中创建ADB表。

```
CREATE TABLE table_test (
 col1 VARCHAR,
 col2 BIGINT,
 col3 BIGINT
) WITH (
 'connector.type' = 'jdbc', -- 使用 jdbc connector
 'connector.dialect' = 'adb', -- 使用connector dialect
 'connector.url' = 'jdbc:mysql://ip:port/db_name', -- jdbc url
 'connector.table' = 'table_name', -- 表名
 'connector.username' = 'xxx', -- ADB账号名
 'connector.password' = 'xxx', -- ADB账号名对应的密码
);
```

更多Flink信息，请参见Flink。

4.日志数据

4.1. 使用Logstash实时采集日志数据

Logstash是一个开源的服务器端数据处理管道，起初用于将日志类数据写入ES中。随着开源社区的不断发展，Logstash可以同时从多个数据源获取数据，并对其进行转换，然后将其发送到您需要的“存储端”。

以日志数据为例，由于AnalyticDB for MySQL支持原生JDBC方式访问，您可以通过开源logstash output插件logstash-output-jdbc将日志数据导入AnalyticDB for MySQL中进行进一步分析。但经过测试发现，在日志量非常大的情况下，通过JDBC方式将数据写入AnalyticDB for MySQL的性能较低，并且非常消耗CPU的资源（JDBC是单条记录写入）。为此，AnalyticDB for MySQL优化了一个基于JDBC的Logstash output plugin插件——logstash-ouput-analyticdb，专门用于以聚合方式向AnalyticDB for MySQL中写入日志数据。

通过logstash-output-analyticdb将数据写入AnalyticDB for MySQL时的性能，相较于logstash-output-jdbc有5倍提升，并且对CPU的消耗也明显降低。

安装

Logstash的安装流程请参见Installing Logstash，以下介绍如何安装logstash-output-analyticdb。

1. 进入logstash根目录：`cd logstash`。
2. 安装logstash-output-analyticdb：`bin/logstash-plugin install logstash-output-analyticdb`。
3. 在logstash目录下创建`vendor/jar/jdbc`目录：`mkdir -p vendor/jar/jdbc`。
4. 将jdbc jar拷贝到`vendor/jar/jdbc`中：`cd vendor/jar/jdbc; wget http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.36/mysql-connector-java-5.1.36.jar`。

至此，已成功安装logstash-output-analyticdb。

使用方式

在config目录下创建一个`logstash-analyticdb.conf`（名字可以自定义）配置文件，`logstash-analyticdb.conf`文件的内容如下所示。

```
input
{
  stdin { }
}
output {
  analyticdb {
    driver_class => "com.mysql.jdbc.Driver"
    connection_string => "jdbc:mysql://HOSTNAME:PORT/DATABASE?user=USER&password=PASSWORD"
    statement => [ "INSERT INTO log (host, timestamp, message) VALUES(?, ?, ?)", "host", "@timestamp", "m
essage" ]
    commit_size => 4194304
  }
}
```

- connection_string：连接AnalyticDB for MySQL的JDBC URL。
- statement：INSERT SQL的声明数组。

更多参数配置：

- max_flush_exceptions：当写入数据出现异常时，设置最大重试次数，默认值100。
- skip_exception：设置是否跳过异常，默认为FALSE，表示出现异常时将重试直到到达最大重试次数 max_flush_exceptions，如果仍然失败，则同步程序抛异常终止。设置为TRUE时，如果达到重试次数后仍是失败，则跳过异常，将异常写入日志。
- flush_size：一次最多攒批数量，和 commit_size 参数搭配使用。
- commit_size：一次最多攒批数据量大小，和 flush_size 参数搭配使用，达到限定值即提交写入任务。

上述配置文件只是一个示例，您需要根据实际业务配置 logstash-analyticdb.conf 文件。与AnalyticDB for MySQL相关的其他配置请参见README。更多logstash配置和使用规则，请参见logstash文档。

至此，配置任务已全部完成，接下来将启动任务。

启动任务

在logstash的安装目录执行 bin/logstash -f config/logstash-analyticdb.conf 启动任务。

注意事项

建议您通过以下命令将logstash升级至最新版本后，再进行数据写入。

```
bin/logstash-plugin update logstash-output-analyticdb
```

4.2. 将日志服务数据投递到ADB

日志作为一种特殊的数据，对处理历史数据、诊断问题等非常重要。对数据分析人员、开发人员或者运维人员而言，日志都是其工作过程中必不可缺的数据来源。阿里云从用户角度出发，支持通过日志服务将Nginx访问日志、Log4j日志、Apache日志以及结构化文本等日志实时同步至ADB，从而使用ADB进行实时日志分析。

前提条件

- 首次使用日志服务时，您需要通过阿里云账号登录日志服务 LOG产品详情页，单击**立即购买**，进入购买页面，然后单击**立即开通**即可购买日志服务，系统自动跳转到日志服务控制台。

  > **说明** 如果您之前已经开通过日志服务，可直接从创建Project开始。

- 在AnalyticDB for MySQL中完成以下准备工作：
  i. 创建AnalyticDB for MySQL集群。
  ii. 创建数据库账号。
  iii. 创建数据库。
  iv. 如果您需要通过外网地址连接AnalyticDB for MySQL集群，请申请外网地址。
  v. 通过CREATE TABLE创建表，存储投递过来的日志数据。

创建投递任务

您可以通过日志服务数据处理模块中的导出功能，将Logstore中采集到的日志投递到AnalyticDB for MySQL。

1.

2. 创建Project和Logstore（日志库）。

   本示例创建adb-test Project和adb-source Logstore，详情请参见创建Project和创建Logstore。

   > **说明** 目前仅支持同地域投递日志，因此日志服务中的Project所属地域应与AnalyticDB for MySQL所属地域相同。

3. 在**日志库**列表中，单击目标Logstore名称前的**> > 数据处理 > 导出 > AnalyticDB**。

   如果您是首次将日志数据投递到AnalyticDB for MySQL，需要为AnalyticDB for MySQL授权，允许AnalyticDB for MySQL访问日志服务。

   i. 单击AnalyticDB后的**+**，系统自动提示您进行授权操作，单击**权限**。

   ii. 在云资源访问授权页面，单击**同意授权**，将角色**AliyunAnalyticDBAccessingLogRole**授予AnalyticDB for MySQL。

4. 完成授权后，单击AnalyticDB后的**+**，您可以选择**直接投递**日志数据或者**前往数据加工**对数据进行处理后再投递。

   > **说明** 本示例选择直接投递数据。

5. 在**LogHub —— 数据投递**页面，按照页面提示进行参数配置。

数据投递参数说明

参数	说明
投递名称	为投递任务取一个名字，便于后续管理。
投递描述	为投递任务填写有意义的描述，便于后续管理。
集群版本	设置AnalyticDB for MySQL集群版本，选择**3.0**。 如何将日志数据投递到AnalyticDB for MySQL 2.0，请参见 通过日志服务同步ECS日志数据到AnalyticDB for MySQL。
集群名称	设置目标AnalyticDB for MySQL集群。
数据库名称	填写目标AnalyticDB for MySQL中的数据库名。
表名	填写目标AnalyticDB for MySQL中的表名。

参数	说明
账号名称	目标AnalyticDB for MySQL中创建的数据库账号： ○ 高权限账号 ○ 普通账号
账号密码	填写账号名称对应的密码。
字段映射	系统自动提取日志服务中最近10分钟的日志字段，同时自动映射对应的AnalyticDB for MySQL字段。
投递时间	设置投递开始时间。 当数据写入日志服务后，日志服务可以实时将数据投递到AnalyticDB for MySQL。
是否过滤脏数据	设置是否过滤脏数据。 ○ **否**：遇到脏数据时，投递任务自动中断。 ○ **是**：遇到脏数据时，过滤掉脏数据。 脏数据包括数据类型转化失败和非空字段为空等。



6. 完成上述参数配置后，单击**确定**，日志服务将在您设置的投递时间将数据实时投递到AnalyticDB for MySQL表中。

查看日志数据

将日志投递到AnalyticDB for MySQL后，您就可以通过SELECT对日志数据进行自由灵活地分析。

管理投递任务

- 监控投递任务：单击投递名称，可以查看投递任务的执行情况，包括投递成功数据量、投递失败数据量以及投递延迟等。

- 修改投递任务：单击**修改投递配置**修改投递任务。

- 重启投递任务：单击**停止**，待投递任务停止后，单击**启动**重启投递任务。

5.消息队列

5.1. 使用Logstash将Kafka数据写入AnalyticDB for MySQL

5.1.1. 概述

Logstash是开源的服务器端数据处理管道，能够同时从多个数据源采集数据，然后对数据进行转换，并将数据写入指定的存储中。AnalyticDB for MySQL完全兼容MySQL，您可以将Logstash Input插件支持的任一数据源中的数据写入AnalyticDB for MySQL。本文介绍如何通过logstash-input-kafka插件将Kafka数据写入到AnalyticDB for MySQL。

Logstash组件介绍

- **输入-采集各种样式、大小和来源的数据**

  在实际业务中，数据往往以各种各样的形式分散或集中地存储在多个系统中，Logstash支持多种数据输入方式，可以在同一时间从多种数据源采集数据。Logstash能够以连续的流式传输方式轻松地从用户的日志、指标、Web应用、数据存储以及AWS服务采集数据。

- **过滤-实时解析和转换数据**

  数据从源传输到目标存储的过程中，Logstash过滤器能够解析各个事件，识别已命名的字段来构建结构，并将它们转换成通用格式，从而更轻松、快速地分析和实现商业价值。

  - 使用Grok从非结构化数据中派生出结构化数据。
  - 从IP地址破译出地理坐标。
  - 将PII数据匿名化，完全排除敏感字段。
  - 简化整体处理，不受数据源、格式或架构的影响

- **输出-导出数据**

  除了AnalyticDB for MySQL以外，Logstash提供多种数据输出方向，灵活解锁众多下游用例。

示例

Kafka是一个高吞吐量的分布式发布、订阅日志服务，具有高可用、高性能、分布式、高扩展、持久性等特点。目前Kafka已经被各大公司广泛使用，同时logstash也可以快速接入业务中，免去重复建设的麻烦。

详情请参见使用Logstash将Kafka数据写入AnalyticDB for MySQL。

5.1.2. 使用Logstash将Kafka数据写入AnalyticDB for MySQL

本文介绍如何使用Logstash将Kafka数据写入AnalyticDB for MySQL。

操作步骤

1. 执行以下命令安装和更新插件。

```
$bin/plugin install
$bin/plugin update
```

logstash从1.5版本开始集成Kafka，logstash 1.5及以上版本中所有插件的目录和命名都发生了改变，插件发布地址为logstash-plugins。

2. 配置插件。
   - **Input配置示例**

     以下配置可以实现对kafka读取端（consumer）的基本使用。

```
input {
  kafka {
    zk_connect => "localhost:2181"
    group_id => "logstash"
    topic_id => "test"
    codec => plain
    reset_beginning => false # boolean (optional)， default: false
    consumer_threads => 5  # number (optional)， default: 1
    decorate_events => true # boolean (optional)， default: false
  }
}
```

     参数说明：

     - group_id：消费者分组，可以通过组ID来指定，不同组之间的消费互不影响，相互隔离。
     - topic_id：指定消费话题（Topic），也可以理解为先订阅某个话题，然后消费。
     - reset_beginning：指定logstash启动后从哪个位置开始读取数据，默认是结束位置，即logstash进程会从上次读取结束时的偏移量开始继续读取数据；如果之前没有消费过，则从头读取数据。

       如果您要导入原数据，需将reset_beginning值改为true，logstash进程将从头开始读取数据，作用类似于cat，但是logstash读到最后一行时不会终止，而是变成tail -F，继续监听相应数据。
     - decorate_events：指定输出消息时会输出自身信息，包括消费消息的大小、opic来源以及consumer的group信息。
     - rebalance_max_retries：当有新的consumer（logstash）加入到同一个group时，将会Reblance，此后将会有Partitions的消费端迁移到新的consumer上。如果一个consumer获得了某个Partition的消费权限，那么它将会向Zookeeper注册Partition Owner registry节点信息，但是有可能此时旧的consumer尚没有释放此节点，此值用于控制注册节点的重试次数。
     - consumer_timeout_ms：在指定时间内没有消息到达将抛出异常，该参数一般无需修改。

     更多Input参数配置请参见Input。

     > **说明** 如果需要多个logstash端协同消费同一个Topic，需要先把相应的Topic分多个Partitions（区），此时多个消费者消费将无法保证消息的消费顺序性，然后把两个或多个logstash消费端配置成相同的group_id和topic_id。

   - **Output配置示例**

```
output {
  jdbc {
    driver_class => "com.mysql.jdbc.Driver"
    connection_string => "jdbc:mysql://HOSTNAME/DATABASE?user=USER&password=PASSWORD"
    statement => [ "INSERT INTO log (host, timestamp, message) VALUES(?, ?, ?)", "host", "@timestamp", "message" ]
  }
}
```

参数说明：

- `connection_string`：AnalyticDB for MySQL的连接地址。
- `statement`：INSERT SQL的声明数组。

更多Output参数配置请参见Output。

3. 在logstash安装目录中执行 `bin/logstash -f config/xxxx.conf` 命令启动任务，将Kafka数据写入AnalyticDB for MySQL。

6.本地数据

6.1. 使用LOAD DATA导入本地数据

本文介绍如何通过LOAD DATA将本地数据导入AnalyticDB for MySQL。

语法

```
LOAD DATA LOCAL '通过该命令导入本地文件'
  INFILE 'file_name' '本地文件的路径，包含文件地址和文件名。'
  [REPLACE | IGNORE] '导入数据遇到主键重复时用当前数据覆盖已有数据或者自动忽略失败行'
  INTO TABLE table_name 'AnalyticDB for MySQL中的目标表名'
  [{FIELDS | COLUMNS}
    [TERMINATED BY 'string'] '定义每行数据的列分隔符，默认为\t，与MySQL一致。'
    [[OPTIONALLY] ENCLOSED BY 'char'] '定义每列数据的enclosed符号'
  ]
  [LINES
    [TERMINATED BY 'string'] '定义行分隔符，默认为\n。'
  ]
  [IGNORE number {LINES | ROWS}] '设置导入数据时忽略开始的某几行'
  [(column_name_or_user_var
  [, column_name_or_user_var] ...)] '设置导入的列，如果不设置，默认按照列的顺序来导入数据'
```

参数

- `LOAD DATA LOCAL INFILE`：表示要进行本地文件导入操作。
- `file_name`：要导入AnalyticDB for MySQL的本地文件的路径，包含文件地址和文件名。

  > **说明** 如果`file_name`使用相对路径，则是相对于客户端程序启动时的路径。

- `table_name`：AnalyticDB for MySQL中的目标表名。

  > **说明** 表名前无需携带数据库名。

- `REPLACE`：导入数据时，遇到主键重复则强制用当前数据覆盖已有数据。
- `IGNORE`：导入数据时，遇到主键重复或者数据问题时则自动忽略失败行，部分行导入失败但不影响其他行导入。
- `[FIELDS] TERMINATED BY 'string'`：定义每行数据的列分隔符，默认为`\t`，与MySQL一致。
- `[FIELDS] ENCLOSED BY 'char'`：定义每列数据的enclosed符号。

  例如，某一列数据为`"a"`，定义`enclosed by '"'`后，导入数据时先将`"a"`前后的`"`移除，然后导入数据。

- [LINES] TERMINATED BY 'string'：定义行分隔符，默认为 \n 。
- IGNORE number LINES：设置导入数据时忽略开始的某几行。例如 IGNORE 1 LINES ，导入数据时忽略第一行数据，即第一行数据不会导入AnalyticDB for MySQL。
- (column_name_or_user_var,...)：设置导入的列，如果不设置，默认按照列的顺序来导入数据。
  - 如果导入列数 > 目标表的列数，则系统自动忽略多余的列。
  - 如果导入列数 < 目标表的列数，则后续缺少的列将自动填充默认值。

注意事项

- 客户端开启 local-file 。

  以mysql-client为例，您需要在 my.cnf 文件中增加以下配置开启 local-file 功能。

  ```
  cat ~/.my.cnf
  [mysqld]
  local-infile
  [mysql]
  local-infile
  ```

  更多 my.cnf 文件信息，请参见MySQL官方文档。
- 导入数据时无法保证操作的原子性。
  - 在 IGNORE 模式下，忽略导入失败的数据行。
  - 在 REPLACE 模式下，一旦有数据行导入失败，系统将中止后续 INSERT 操作，因此可能存在部分行数据导入，部分行数据未导入的情况。
- 支持通过 SHOW WARNINGS 命令，查看失败行的错误信息。

示例

本示例将本地文件 out.bak 中的数据导入AnalyticDB for MySQL的 test 表中。 out.bak 文件中共有5000行数据，列分隔符为 \t ，行分隔符 \n ，其中第10行数据存在问题，如下所示。

```
1 bb
2 bb
3 bb
4 bb
5 bb
6 bb
7 bb
8 bb
9 bb
bb  10
...
```

1. 连接AnalyticDB for MySQL集群，通过CREATE DATABASE和CREATE TABLE，在 `adb_demo` 数据库下创建表 `test` 表，从本地文件导入的数据将存储在 `test` 表中。

```
CREATE TABLE test (
a int(11) NOT NULL DEFAULT '0',
b varchar(8) NOT NULL,
PRIMARY KEY (a)
) DISTRIBUTED by hash(a);
```

2. 在mysql-client中执行LOAD DATA命令将本地文件 `out.bak` 中的数据导入AnalyticDB for MySQL的 `test` 表中。

```
#执行LOAD DATA命令， ignore模式下，部分行导入失败但不影响其他行导入。
mysql> load data local infile '~/out.bak' ignore into table test FIELDS TERMINATED BY '\t' LINES TERMIN
ATED BY '\n' ;
Query OK, 4999 rows affected, 1 warning (0.69 sec)
mysql> show warnings;
+---------+-------+------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| Level   | Code  | Message
|
+---------+-------+------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| Warning | 13000 | com.alibaba.cloud.analyticdb.common.sql.hardcode.HardParserException: syntax er
ror :syntax error => IDENTIFIER is not value type pos:33 row: 0 and ceil:0 |
+---------+-------+------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
1 row in set (0.01 sec)
mysql> select count(1) from test;
+----------+
| count(1) |
+----------+
|     4999 |
+----------+
1 row in set (0.14 sec)
#执行LOAD DATA命令，replace模式下，部分行导入失败后立即终止后续导入操作。
mysql> load data local infile '~/out.bak' replace into table test FIELDS TERMINATED BY '\t' LINES TERMIN
ATED BY '\n' ;
ERROR 1064 (42000): [13000, 2019061210070703000511314203303000266] syntax error :syntax error => I
DENTIFIER is not value type pos:34 row: 0 and ceil:0
#执行LOAD DATA命令，导入数据时会跳过前10条数据。
mysql> load data local infile '~/out.bak' replace into table test FIELDS TERMINATED BY '\t' LINES TERMIN
ATED BY '\n' IGNORE 10 LINES;
Query OK, 4990 rows affected (0.37 sec)
mysql> show warnings;
Empty set (0.00 sec)
```

7.异步提交导入导出任务

AnalyticDB for MySQL中支持通过异步方式提交数据导入导出任务。

异步提交任务

- **语法**

```
submit job insert overwrite into xxx select ...
```

执行上述命令后，将返回一个job_id。

- **示例**

```
mysql> submit job insert overwrite into test select * from test_external_table;
+----------------------------------------+
| job_id                                 |
+----------------------------------------+
| 2017112122202917203100908203303000715 |
```

查询异步任务状态

- **语法**

```
show job status where job= ‘job_id’
```

- **示例**

```
mysql> show job status where job='2017112122202917203100908203303000715';
+----------------------------------------+-------------+---------+----------+----------------------+----------------------+----------------------+
| job_id                                 | schema_name | status  | fail_msg | create_time          | update_time          | definition           |
+----------------------------------------+-------------+---------+----------+----------------------+----------------------+----------------------+
| 2017112122202917203100908203303000715 | test        | RUNNING | NULL     | 2017-11-21 22:20:31.0 | 2017-11-21 22:20:40.0 | insert into test select * from test |
```

- **任务状态说明**
  - `INIT` ：任务进入队列
  - `RUNNING` ：后台开始执行任务
  - `FINISH` / `FAILED` ：任务成功或失败

终止任务

- **语法**

```
cancel job 'job_id'
```

- **示例**

```
mysql> cancel job '20171121222029172031009082033030007l5';
```

- **说明**
  - 未调度起来的任务和已完成（失败或成功）的任务会被移除队列。
  - 正在运行的任务会被终止，已导入的数据也会被回滚。