ALIBABA CLOUD

阿里云

数据湖分析

SQL 参考

文档版本：20201110

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.常用SQL ---- 05
2.数据源与功能的矩阵 ---- 06
3.Hints ---- 07
4.注意事项和限制 ---- 12
5.支持的文件格式 ---- 13
6.Table Properties ---- 20
7.性能优化指南 ---- 23

1.常用SQL

本文主要介绍DLA中常用的SQL类型。

DLA SQL是基于 Presto 构建，DDL部分是兼容 Hive的DDL标准， DML部分是支持ANSI SQL 语法。

- 数据类型 ：目前DLA支持的数据类型
- Hint： 添加Hint可以干预SQL的执行，比如控制并发等。
- 支持的文件格式：对于OSS数据源或者类似的文件系统支持的数据格式。
- Table Properties：新建表时支持的属性，可以设置压缩格式、列名映射等。
- 支持数据源与功能的矩阵：支持的数据源及功能矩阵，包括常见的OSS、RDS等。
- 函数说明：DLA SQL是基于开源227版本的Presto改进的，支持大部分开源227版本的函数，具体可以参考开源 Presto的函数。
- DDL
  - 常见DDL
  - CREATE SCHEMA：创建SCHEMA/DATABASE
  - CREATE TABLE：创建表
  - DROP SCHEMA：删除SCHEMA/DATABASE
  - DROP TABLE：删除表
  - ALTER TABLE：更改表的结构及分区信息
  - 针对源端为OSS的元数据操作
  - MSCK REPAIR TABLE：同步OSS数据源上实际的数据分区信息到元数据分区中
  - MSCK REPAIR TABLE SYNC_DIR：同步OSS数据源 一个目录的分区信息到元数据分区中
  - 针对源端为数据库的元数据操作
  - MSCK REPAIR DATABASE：自动 关联 源数据库的所有表
  - CREATE TABLE LIKE MAPPING：自动根据 源端的表的结构 推断表结构
  - 常见的查询DDL
  - SHOW SCHEMAS：查询用户所有的SCHEMA/DATABASE
  - SHOW TABLES：查询用户当前SCHEMA下的表
  - SHOW CREATE TABLE：查看建表语句
  - SHOW PARTITIONS：列出表的所有分区信息
  - SHOW QUERY_TASK：查询用户的查询任务信息
- DML
  - INSERT：插入数据
  - SELECT：查询
- ACL
  - GRANT：为账号授权
  - REVOKE： 撤销账号权限

2.数据源与功能的矩阵

DLA支持云上大部分的数据源，相关支持的数据源不断优化，不支持的点可以提交工单，DLA团队将会快速响应；

RDS包括 MySQL、PG、SQLServer、Oracle关系型数据库，PolarDB跟RDS保持一致；

Data Source\ Feature	SELECT	INSERT INTO	INSERT OVERWRITE	CREATE TABLE LIKE MAPPING	MSCK REPAIR DATABASE	MSCK REPAIR TABLE	ADD COLUMN
OSS	✓	✓	✓	✓	✓	✓	✓
RDS	✓	✓	✗	✓	✓	✗	✗
OTS	✓	✓	✗	✗	✓	✗	✓
ODPS	✓	✓	✗	✓	✓	✗	✗
ADB-MySQL 2.0	✓	✓	✗	✗	✓	✗	✗
ADB-MySQL 3.0	✓	✓	✓	✗	✓	✗	✗
ADB-PG	✓	✓	✓	✗	✓	✗	✗
MongoDB	✓	✓	✗	✗	✗	✗	✗
Redis	✓	✓	✗	✓	✓	✗	✗

3.Hints

DLA的查询默认是同步执行的，也就是说客户端到DLA的连接会一直等待服务端，知道查询结果返回。但是对于一些大规模的ETL，比如 `INSERT INTO SELECT FROM` 这类语句，如果数据量大的话，它的执行时间是很长的，服务端长时间不返回任何数据给客户端，连接可能会中断，因此就有了“异步执行”的概念。异步执行的场景下，用户提交一个SQL，服务端马上返回一个ID，后续用户可以根据这个ID来查询SQL的执行情况。

```
mysql -h<end_point_host> -P<end_point_port>  -u<user_name> -p<password> -c -A
.....................
mysql> use public_dataset_tpch_1x_text;
Database changed
mysql> /*+ run-async=true */select count(*) from customer;
+--------------------------------+
| ASYNC_TASK_ID                  |
+--------------------------------+
| q202006161430hz89aab7ef0245339 |
+--------------------------------+
1 row in set (0.04 sec)
```

MySQL client 默认发送到服务器的 SQL statements 中删除 comments（包括优化器提示），直到 MySQL 5.7.7。如果使用小于5.7.7版本，请使用 -c 选项 支持添加 comments（包括优化器提示） 提示

通用类

run-async

- 含义： 是否异步执行。
- 取值范围： true, false, 默认false。

max-running-time-sec

- 含义： 查询最大执行的时间，单位是秒。
- 取值范围： 大于0的整数，时间不要超过6个小时。

这个hint可以控制SQL执行的时间，如果超过这个时间，SQL就被系统自动终止掉。

join-distribution-type

- 含义： 表在JOIN的时候使用的数据分发类型。
- 取值范围： AUTOMATIC/BROADCAST/PARTITIONED
- 默认值： PARTITIONED

如果右表比较小能够放进内存，采用BROADCAST会极大的提高查询的性能；而PARTITIONED没有这个限制，所以可以支持更大的JOIN，但是对于小表JOIN性能较差。

pool-selector

- 含义： 指定查询要发送到的资源池。

- 取值范围： -

OSS

insert-overwrite-ignore-conflict

- 含义： 执行insert overwrite时，如果写入的目标目录已经存在，是否强制覆盖。如果指定为false，则只会覆盖元数据中有记录的目标记录；如果为true，则无论如何都会覆盖。
- 取值范围： true/false
- 默认值： false

JDBC

jdbc-scan-splits

- 含义： 控制在查询JDBC类数据源时，拆分多少个split来获取数据
- 取值范围： 1 - 500
- 默认值： 1

jdbc-split-column

- 含义： 查询JDBC类数据源时，选择哪一列作为切分列，注意，切分列的类型只支持INT/BIGINT和带索引的CHAR/VARCHAR类型，如果选择的切分列不属于上述类型，执行引擎会自动忽略这个参数。
- 取值范围： 列名
- 默认值： 无

ots-query-version

- 含义： 是否启动V2新版本协议来查询
- 取值范围： 1, 2
- 默认值： 2

ots-filter-version

- 含义： 是否启动V2新版本的filter来查询
- 取值范围： 1, 2
- 默认值： 2

ots-split-unit-mb

- 含义： split默认的切分单位
- 取值范围： 1, 2
- 默认值： 2

ots-fetch-size

- 含义：每次请求 TableStore 返回的数据行数
- 取值范围：100，100000
- 默认值：10000

ots-start-version

- 含义：数据版本起点，是一个时间戳（millisecond），为了快速过滤掉大量的多版本数据
- 默认值： -1

ots-end-version

- 含义：数据版本终点，是一个时间戳（millisecond），为了快速过滤掉大量的多版本数据
- 默认值： -1

ots-split-optimize

- 含义： 是否启动split的分组重排等优化逻辑
- 取值范围： true/false
- 默认值： false

ots-split-size-ratio

- 含义： 启动split的分组重排等优化逻辑之后，希望split的数量
- 取值范围： 0.0001 - 1.0
- 默认值： 0.5

ots-partition-prune

- 含义： 是否启动split的范围再做裁剪优化，加快性能
- 取值范围： true/false
- 默认值： true

ots-insert-as-update

- 含义： 是否使用update功能来替换insert
- 取值范围： true/false
- 默认值： false

ots-loose-cast

- 含义： 是否允许使用宽松的cast（比如long值到double，double到long）
- 取值范围： true/false
- 默认值： false

ots-index-first

- 含义： 是否对于某些满足条件的表做索引优先查询。
- 取值范围
  - auto： 会寻找与表相关的索引，只要有满足条件的索引，就会强制使用
  - custom： 根据用户选择表列表，来自动选择满足条件的索引；其中tbl1不需要显示指定库名，是因为当前连接上已经绑定了一个库（比如use xxx）；如下case中，只有tbl1和tbl2会走索引，而tbl3则不会：

```
/*+ ots-index-first=[tbl1, dla_schema2.tbl2, ...] */ select * from tbl1
join dla_schema2.tbl2 join dla_schema3.tbl3 where ...
```

- threshold： 根据当前条件匹配的数据量来动态决策，如果找到一个索引，其匹配的数据量小于一定的行数或者一定比例，那就会自动选择；threshold：200表示where条件匹配的行数不超过200行才会使用，而threshold：5%则表示匹配的比例不超过5%才会使用（至于200和5%，DLA内部会调用Table Store的count接口做快速测试并预估判断）

```
/*+ ots-index-first=threshold：200 */ select * from tbl1 where ...
/*+ ots-index-first=threshold：5% */ select * from tbl1 where ...
```

- 默认值： 不使用索引

ots-index-parallel-scan-mode

- 含义：是否使用 TableStore 的并发导出功能。
- 取值范围：true, false
- 默认值：false （默认情况下不使用并发导出功能）

更多详情以及注意事项请参见：https://developer.aliyun.com/article/776638?groupCode=datalakanalytics

消息通知

DLA支持对SQL在执行完成之后进行异步通知，目前支持两种通知渠道，对应的hint组合如下：

ONS

```
/*+ run-async=true, mq-notify-by=ons, mq-topic=${您的ons的topic},
mq-producer-id=${您的group Id}, mq-endpoint=${您的某个endpoint，与DLA所在region相同} */
```

- mq-notify-by： 消息渠道的名称
- mq-topic： ons的topic
- mq-producer-id： ons的group Id
- mq-endpoint： ons的endpoint

MNS

```
/*+ run_async=true, mq-notify-by=mns, mq-queue=${您的mns队列} */
```

- mq-notify-by： 消息渠道的名称
- mq-queue： 您的mns队列

异步查询结果导出

当你使用异步执行的时候，执行的结果会被导出到OSS上，默认保存的CSV格式的，我们也提供了一些hint可以对保存的格式进行调整：

```
/*+ force-persist-result=true, result-file-format=csv, result-col-del=[,],
result-row-del=\r\n, result-meta-included=true */
select * from tbl1 ...
```

异步导出到OSS是写入到OSS是单线程执行，如果数据量较大，执行速度会很慢；如果导出大量数据，建议新建一个OSS的表，再写入；

- result-file-format： 目前支持csv/json。
- result-col-del： 列分隔符（只有在格式为CSV的时候才有作用）
- result-row-del： 行分隔符（只有在格式为CSV的时候才有作用）
- result-meta-included： 是否要在第一行输出列名称（默认是不包含的）
- result-oss-location： 结果文件的OSS目录。

很多客户希望用’，’而不是’|’作为分隔符，因为’，’本身有特殊语义，目前通过[,]来转义，比如 `result-col-del=[,], result-row-del=\r\n`

有时候我们使用同步查询的时候也想把结果保存一份到OSS上，可以用这个hint：

- force-persist-result=true

4.注意事项和限制

限制项目	上限
DLA控制台SQL查询结果条数	400
用户库的个数	4096
用户单库表的个数	4096
用户每个表的partition个数	60000
单表列的个数	4096
SQL执行器最高的并发度	400
SQL拆解stage个数	120
单UID最大连接数	10
单SQL的最大长度（字节）	16777215

5.支持的文件格式

这篇文档介绍DLA支持的文件的格式。

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_avro (
  L_ORDERKEY INT,
  L_PARTKEY INT,
  L_SUPPKEY INT,
  L_LINENUMBER INT,
  L_QUANTITY DOUBLE,
  L_EXTENDEDPRICE DOUBLE,
  L_DISCOUNT DOUBLE,
  L_TAX DOUBLE,
  L_RETURNFLAG STRING,
  L_LINESTATUS STRING,
  L_SHIPDATE DATE,
  L_COMMITDATE DATE,
  L_RECEIPTDATE DATE,
  L_SHIPINSTRUCT STRING,
  L_SHIPMODE STRING,
  L_COMMENT STRING
)
STORED AS AVRO
LOCATION 'oss://bucket001/datasets/test/test_avro/';
```

ORC

```
CREATE EXTERNAL TABLE test_orc (
  L_ORDERKEY INT,
  L_PARTKEY INT,
  L_SUPPKEY INT,
  L_LINENUMBER INT,
  L_QUANTITY DOUBLE,
  L_EXTENDEDPRICE DOUBLE,
  L_DISCOUNT DOUBLE,
  L_TAX DOUBLE,
  L_RETURNFLAG STRING,
  L_LINESTATUS STRING,
  L_SHIPDATE DATE,
  L_COMMITDATE DATE,
  L_RECEIPTDATE DATE,
  L_SHIPINSTRUCT STRING,
  L_SHIPMODE STRING,
  L_COMMENT STRING
)
STORED AS ORC
LOCATION 'oss://bucket001/datasets/test/test_orc';
```

Paruqet

```
CREATE EXTERNAL TABLE test_parquet_hive_serde (
  L_ORDERKEY INT,
  L_PARTKEY INT,
  L_SUPPKEY INT,
  L_LINENUMBER INT,
  L_QUANTITY DOUBLE,
  L_EXTENDEDPRICE DOUBLE,
  L_DISCOUNT DOUBLE,
  L_TAX DOUBLE,
  L_RETURNFLAG STRING,
  L_LINESTATUS STRING,
  L_SHIPDATE DATE,
  L_COMMITDATE DATE,
  L_RECEIPTDATE DATE,
  L_SHIPINSTRUCT STRING,
  L_SHIPMODE STRING,
  L_COMMENT STRING
)
STORED AS PARQUET
LOCATION 'oss://bucket001/datasets/test/test_parquet_hive_serde';
```

RcFile

```
CREATE EXTERNAL TABLE test_rcfile (
  L_ORDERKEY INT,
  L_PARTKEY INT,
  L_SUPPKEY INT,
  L_LINENUMBER INT,
  L_QUANTITY DOUBLE,
  L_EXTENDEDPRICE DOUBLE,
  L_DISCOUNT DOUBLE,
  L_TAX DOUBLE,
  L_RETURNFLAG STRING,
  L_LINESTATUS STRING,
  L_SHIPDATE DATE,
  L_COMMITDATE DATE,
  L_RECEIPTDATE DATE,
  L_SHIPINSTRUCT STRING,
  L_SHIPMODE STRING,
  L_COMMENT STRING
)
STORED AS RCFILE
LOCATION 'oss://bucket001/datasets/test/test_rcfile';
```

TextFile: Normal

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_text_null (
  URL STRING,
  TITLE STRING
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/datasets/test/test_null/textfile/';
```

TextFile: OpenCSV

OpenCSVSerde在使用时需要注意以下几点：

- 用户可以为行的字段指定字段分隔符、字段内容引用符号和转义字符，例如：WITH SERDEPROPERTIES（“separatorChar” = “,”，“quoteChar” = “`”，“escapeChar” = “\”);
- 不支持字段内嵌入的行分割符；
- 所有字段定义STRING类型；
- 其他数据类型的处理，可以在SQL中使用函数进行转换。

```
CREATE EXTERNAL TABLE test_opencsv (
 id STRING,
 name STRING,
 location STRING,
 create_date STRING,
 create_timestamp STRING,
 longitude STRING,
 latitude STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
with serdeproperties(
"separatorChar"=",",
"quoteChar"="\"",
"escapeChar"="\\"
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/datasets/test/test_opencsv';
```

TextFile: MultiDelimitSerDe

这个Serde的主要特点是对于TextFile里面列分隔符可以使用多个字符来做分隔。

```
CREATE EXTERNAL TABLE test_csv_multidelimit (
 id STRING,
 name STRING,
 location STRING,
 create_date STRING,
 create_timestamp STRING,
 longitude STRING,
 latitude STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.MultiDelimitSerDe'
with serdeproperties(
"field.delim"="||"
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/datasets/test/test_multidelimiter';
```

RegexSerDe

这个Serde的主要使用场景是对一些没有固定格式、需要借助正则表达式来对数据中的字段进行扣取的场景：

```
CREATE EXTERNAL TABLE IF NOT EXISTS test_regex (
 host STRING,
 identity STRING,
 `user` STRING,
 time STRING,
 request STRING,
 status STRING,
 size INT,
 referer STRING,
 agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
 "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) (-|\\[[^\\]]*\\]) ([^ \"]*|\"[^\"]*\") (-|[0-9]*) (-|[0-9]*)(?: ([^ \"]*|\"[^\"]*\") ([^ \
"]*|\"[^\"]*\"))?"
)
STORED AS TEXTFILE
LOCATION 'oss://oss-cn-beijing-for-openanalytics-test/datasets/test/test_regex';
```

JSON: org.apache.hadoop.hive.serde2.JsonSerDe

```
CREATE EXTERNAL TABLE IF NOT EXISTS `customer_case_jiahe`.`single_latin1_broken` (
  `id` int ,
  `name` string ,
  `age` int
)
STORED AS JSON
LOCATION 'oss://oss-cn-beijing-for-openanalytics-test/datasets/test/customer_case/jiahe/single_latin1_en
code_broken_record.log';
```

JSON: org.apache.hive.hcatalog.data.JsonSerDe

```
CREATE external TABLE json_table_1 (
  docid string,
  user_1 struct<
        id:INT,
        username:string,
        name:string,
        shippingaddress:struct<
                    address1:string,
                    address2:string,
                    city:string,
                    state:string
                    >,
        orders:array<
              struct<
                 itemid:INT,
                  orderdate:string
                  >
               >
        >
  )
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION 'oss://oss-cn-beijing-for-openanalytics-test/datasets/test/json/hcatalog_serde/table_1/test_json
.json';
```

JSON EsriJsonSerDe

DLA支持Esri ArcGIS的地理JSON数据文件的SerDe处理，关于这种地理JSON数据格式说明，可以参考：https://github.com/Esri/spatial-framework-for-hadoop/wiki/JSON-Formats

```
CREATE EXTERNAL TABLE IF NOT EXISTS california_counties
(
 Name string,
 BoundaryShape binary
)
ROW FORMAT SERDE 'com.esri.hadoop.hive.serde.EsriJsonSerDe'
STORED AS INPUTFORMAT 'com.esri.json.hadoop.EnclosedEsriJsonInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 'oss://oss-cn-beijing-for-openanalytics-test/datasets/geospatial/california-counties/'
```

6.Table Properties

我们各种不同的表支持不同的table properties，如下

table_mapping

当DLA里面的表名跟底层数据与的表名不一致的时候，可以用这个property来指定底层数据与的表名，比如我们DLA的表名是 `person` , 而底层的表名叫 `staff` , 那么可以这么写：

```
create external table person (
  id int,
  name string,
  age int
) tblproperties(
 table_mapping = 'staff'
);
```

column_mapping

当DLA里面的列名跟底层数据与的列名不一致的时候，或者底层压根就没有列的概念的时候，可以用这个property来指定底层数据与DLA列名的对应关系。

比如下面这个例子：

```
create external table person (
  id int,
  name string,
  age int
) tblproperties(
 column_mapping = 'id,identifier;name,title;'
);
```

DLA的列名 `id` 对应到了底层的 `identifier` , `name` 对应到了 `title` , 而没有指定的 `age` 则对应到底层的 `age` 。

OSS

auto.create.location

`auto.create.location` 的作用是我们可以指定一个OSS上一个不存在的目录来建表，而DLA会自动创建OSS上对应的目录。比如：

```
CREATE EXTERNAL TABLE person (
  `id` int,
  `name` string,
  `age` int
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/dir001/'
TBLPROPERTIES (
  'auto.create.location'='true'
);
```

compression.type

使用DLA写入OSS的数据默认的压缩格式是SNAPPY，如果不喜欢可以使用 `compression.type` 手动指定压缩格式：

```
CREATE EXTERNAL TABLE person (
  `id` int,
  `name` string,
  `age` int
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/dir001/'
TBLPROPERTIES (
  'compression.type'='gzip'
);
```

目前支持的压缩格式有：

- SNAPPY
- GZIP
- NONE(不压缩)

directory.odps

当需要使用DLA分析MaxCompute的OSS外表数据时，需要指定这个参数：

```
CREATE EXTERNAL TABLE person (
  `id` int,
  `name` string,
  `age` int
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/dir001/'
TBLPROPERTIES (
  'directory.odps'='true'
);
```

这是因为MaxCompute外表的目录结构和普通OSS不同，如果不指定这个参数会导致找不到数据文件。

skip.header.line.count

`skip.header.line.count` 针对文本类型的数据，指定要跳过的行数，比如有些CSV文件第一行是文件头，不是真正的数据，分析的时候需要跳过。

```
CREATE EXTERNAL TABLE person (
  `id` int,
  `name` string,
  `age` int
)
STORED AS TEXTFILE
LOCATION 'oss://bucket001/dir001/'
TBLPROPERTIES ("skip.header.line.count"="1");
```

7.性能优化指南

对于JDBC类数据源，比如MySQL/SQLServer/PostgreSQL/AnalyticDB/AdbPG/Druid等等，我们拉取数据的时候默认是单线程拉取，这个在大多数场景是足够使用的，但如果你需要加快读取性能，可以通过如下方法加速：

- 如果您的底层RDS性能比较好，你可以通过 `jdbc-splits-scan` 来采用多个线程并发拉数据。

```
/*+jdbc-scan-splits=5*/ select * from tbl1;
```

- 执行引擎默认会选择每一列作为切分列，如果你希望自己指定切分列，可以通过jdbc-split-column指定，需要注意的是，切分列的类型只支持INT/BIGINT和带索引的CHAR/VARCHAR类型，如果选择的切分列不属于上述类型，执行引擎会自动忽略这个参数。

```
/*+jdbc-split-column=columnName1*/ select * from tbl1;
```