数据湖计算 DLC

客户端访问

腾讯云

文档目录

客户端访问
TDLC 命令行工具访问

最近更新时间：2024-04-09 09:58:41

TDLC 是腾讯云数据湖计算 (DataLake Compute，DLC) 提供的客户端命令工具。通过 TDLC 工具，您可以向 DLC 数据引擎 提交 SQL、Spark 任务。
TDLC 使用 Go 编写，基于 Cobra 框架，支持配置多个存储桶和跨桶操作。您可以通过 `./tdlc [command] --help` 来查看 TDLC 的使用方法。

下载与安装

TDLC 命令行工具提供 Windows、Mac、Linux 操作系统的二进制包，通过简单的安装和配置后即可使用。您可以根据客户端的操作系统类型选择下载。

操作系统	TDLC 二进制包 下载地址
Windows	tdlc-windows-v0.1.1
Mac	tdlc-macos-v0.1.1
Linux	tdlc-linux-v0.1.1

将下载的文件重命名为 `tdlc` 。打开客户端的命令行，切换到下载路径下，如果您是Mac/Linux系统，需要使用 `chmod +x tdlc` 命令授予文件可执行权限。执行 `./tdlc` 后，成功展示如下内容即安装成功可以使用。

```
Tencentcloud DLC command tools is used to play around with DLC.
With TDLC user can manger engines, execute SQLs and submit Spark Jobs.

Usage:
  tdlc [flags]
  tdlc [command]

Available Commands:
  config
  help        Help about any command
  spark       Submit spark app to engines.
  sql         Executing SQL.
  version

Flags:
      --endpoint string     Endpoint of Tencentcloud account. (default
"dlc.tencentcloudapi.com")
```

```
      --engine string       DLC engine. (default "public-engine")
  -h, --help                help for tdlc
      --region string       Region of Tencentcloud account.
      --role-arn string     Required by spark jar app.
      --secret-id string    SecretId of Tencentcloud account.
      --secret-key string   SecretKey of Tencentcloud account.
      --token string        Token of Tencentcloud account.

Use "tdlc [command] --help" for more information about a command.
```

使用说明

全局参数

TDLC 提供如下全局参数。

全局参数	说明
--endpoint string	服务连接地址，默认使用 dlc.tencentcloudapi.com
--engine string	DLC 数据引擎名称，默认值为 public-engine。建议您使用独享数据引擎
--region string	使用地域。如 ap-nanjing, ap-beijing, ap-guangzhou，ap-shanghai，ap-chengdu，ap-chongqing, na-siliconvalley, ap-singapore, ap-hongkong
--role-arn string	当您提交 spark 作业时，需要指定访问 cos 文件的权限，此次指定权限角色的 rolearn，rolearn 详情可以参考 配置数据访问策略 。
--secret-id string	腾讯云账号的 secretId
--secret-key string	腾讯云账号的 secretKey
--token string	（选填）腾讯云账号临时 token

CONFIG 命令

config 可以配置常用的参数，配置的参数会以默认值提供。命令行参数会覆盖已配置 config 的参数。

命令	说明
list	列出当前的默认配置

set	变更配置
unset	重置配置

示例：

```
./tdlc config list
./tdlc config set secret-id={1} secret-key={2} region={b}
./tdlc config unset region
```

SQL 子命令

SQL 子命令目前仅支持 Presto 或者 SparkSQL 集群，以下是 SQL 子命令支持的参数。

参数	说明
-e, --exec	执行 SQL 语句
-f, --file	执行 SQL 文件，如果有多个SQL文件请用 ; 分割
--no-result	执行后不获取结果
-p, --progress	显示执行进度
-q, --quiet	安静模式，提交任务后不等待任务执行状态

示例：

```
./tdlc sql -e "SELECT 1" --secret-id aa --secret-key bb --region ap-beijing --engine
public-engine
./tdlc sql -f ~/biz.sql --no-result
```

SPARK 子命令

Spark 子命令包含以下命令，可以用以提交 Spark 作业、查看运行日志、终止任务。

命令	说明
submit	通过 spark-submit 提交任务
run	执行 spark 作业

log	查看运行日志
list	查看 spark 作业列表
kill	终止任务

以下是 Spark submit 子命令 支持的参数，列表中文件相关参数支持使用本地文件或 COSN 协议。

参数	说明
--driver-size	driver规格，默认使用 small、medium、large、xlarge，内存型集群使用 m.xmall、m.medium、m.large、m.xlarge
--executor-size	executor 规格，默认使用 small、medium、large、xlarge，内存型集群使用 m.xmall、m.medium、m.large、m.xlarge
--executor-num	executor 数量
--files	查看 spark 作业列表
--archives	依赖压缩文件
--class	Java/Scala 运行的主函数
--jars	依赖的jar包，使用 , 分割
--name	程序名称
--py-files	依赖的python文件，支持.zip、.egg、.py格式
--conf	额外配置

示例：

```
./tdlc spark submit --name spark-demo1 --engine sparkjar --jars /root/sparkjar-dep.jar --class com.demo.Example /root/sparkjar-main.jar arg1
./tdlc spark submit --name spark-demo2 cosn://bucket1/abc.py arg1
```

第三方软件联动

最近更新时间：2023-07-26 15:28:22

腾讯云数据湖计算 DLC 在产品定位上秉承敏捷、开放的原则，支持大量的第三方软件集成，包括调度工具、交互式开发工具、BI 工具等。目前也在不断地支持和测试其他第三方软件集成。

说明

- 以下列出的第三方软件，DLC 产研团队测试了主流版本的核心功能，并未能覆盖全部版本号。
- 如果您在第三方软件集成 DLC 的使用上有问题，或者有其他第三方软件集成的需求，欢迎您 提交工单 联系我们。
- 如果有数据集成和调度的需求，欢迎使用数据开发治理平台WeData 。

调度工具

如您已经部署/拥有如下第三方软件，可以将其连接到 DLC，管理和调度 DLC 上的作业。

第三方软件	说明
Apache Airflow	Airflow 是一个工作流管理平台，可以用于管理调度和执行。
Apache DolphinScheduler	DolphinScheduler 是一个可视化 DAG 工作流任务调度平台。

交互式计算

如您已经部署/拥有如下第三方软件，可以将其连接到 DLC，使用 DLC 的能力进行计算和分析。

第三方软件	说明
Yanagishima	Yanagishima 是一个用以可视化访问 Trino，Hive，Spark 的开源 Web 应用程序。
Apache Zeppelin	Zeppelin 是一个基于 Web 的交互式计算环境。
Jupyter Notebook	Jupyter Notebook 是一个基于 Web 的交互式计算平台。

数据库工具

如您已经部署/拥有如下第三方数据库工具，可以将其连接到 DLC，查询 DLC 数据。

第三方软件	说明
SQL Workbench	SQL Workbench 是一个跨平台 SQL 查询工具。您可以使用您自己部署的 SQL Workbench 访问到 DLC。
DBeaver	DBeaver 是一个通用数据库工具。

商业智能

如您已经部署/拥有如下第三方商业智能软件，可以将其连接到 DLC，进行商业智能分析并输出报表，辅佐您的商业决策。

第三方软件	说明
Metabase	Metabase 是一款易用的报表系统。您可以使用自己部署的 Metabase 访问到 DLC 生成报表。
Redash	Redash 是一款开源 BI 工具，提供了基于 Web 的数据库查询和数据可视化功能。
FineBI	FineBI 是帆软软件有限公司推出的一款商业智能产品。
Tableau	Tableau 是一个可视化分析平台，提供强大、安全且灵活的端到端分析平台，提供从连接到协作的一整套功能。
CBoard	CBoard 是由上海楚果信息技术有限公司主导开源的一款自助 BI 数据分析产品。
Apache Superset	Apache Superset 是一个现代化的商业智能网络应用程序，由 Airbnb 开源。

JDBC 访问

最近更新时间：2023-12-22 16:32:35

环境准备

- 依赖：JDK 1.8
- JDBC 下载：点击下载 JDBC 驱动

连接 DLC

1. 加载 DLC JDBC 驱动

```
Class.forName("com.tencent.cloud.dlc.jdbc.DlcDriver");
```

2. 通过 DriverManager 创建 Connection

```
Connection cnct = DriverManager.getConnection(url, secretId, secretKey);
```

url 格式

```
jdbc:dlc:<dlc_endpoint>?
task_type=SQLTask&database_name=abc&datasource_connection_name=DataLakeC
atalog&region=ap-nanjing&data_engine_name=spark-
cu&result_type=COS&read_type=Stream
```

JDBC 链接串参数说明：

参数	必填	说明
dlc_endpoint	是	DLC 服务的 Endpoint，固定为 dlc.tencentcloudapi.com
datasource_connection_name	是	数据源连接名称，对应 DLC 的数据目录
task_type	是	任务类型。 Presto 引擎填写：SQLTask SparkSQL 引擎填写：SparkSQLTask Spark 作业引擎填写：BatchSQLTask
database_name	否	数据库名称

region	是	地域，目前 DLC 服务支持 ap-nanjing, ap-beijing, ap-guangzhou，ap-shanghai， ap-chengdu，ap-chongqing, na-siliconvalley, ap-singapore, ap-hongkong
data_engine_name	是	数据引擎名称
secretId	是	腾讯云 API 密钥管理中的 SecretId
secretKey	是	腾讯云 API 密钥管理中的 Secretkey
result_type	否	默认为 Service。如果您对获取结果的速度有更高要求，可以使用设置为 COS。 Service：通过 DLC 接口获取结果 COS：通过 COS 客户端获取结果
read_type	否	Stream：流式从 COS 获取结果 DownloadSingle：单个文件下载到本地获取结果 默认值为 Stream.只有当 result_type 为 COS，该值才有意义。 下载文件位置为/tmp 临时目录，请确保该目录有读写权限，数据读取完成会自动删除。

执行查询

```
Statement stmt = cnct.createStatement();ResultSet rset =
stmt.executeQuery("SELECT * FROM dlc");
while (rset.next())
 {// process the results
}
rset.close();
stmt.close();
conn.close();
}
rset.close();
stmt.close();
conn.close();
```

语法支持

目前 jdbc 可以使用的语法与 DLC 标准语法保持一致。

实例代码

库表操作

```
import java.sql.*;
public class MetaTest {
      public static void main(String[] args) throws SQLException {
         try {
            Class.forName("com.tencent.cloud.dlc.jdbc.DlcDriver");
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
         return;
   }
   Connection connection = DriverManager.getConnection(
               "jdbc:dlc:<dlc_endpoint>?task_type=<task_type>&database_name=
<database_name>&datasource_connection_name=DataLakeCatalog&region=
<region>&data_engine_name=<data_engine_name>&result_type=<result_type>",
               "<secret_id>",
               "secret_key");
   Statement statement = connection.createStatement();
   String dbName = "dlc_db1";
   String createDatabaseSql = String.format("CREATE DATABASE IF NOT EXISTS %s",
dbName);
   statement.execute(createDatabaseSql);
   String tableName = "dlc_t1";
   String wholeTableName = String.format("%s.%s", dbName, tableName);
   String createTableSql =
   String.format(
   "CREATE EXTERNAL TABLE %s (  "
         + "  id string ,   "
         + "  name string ,   "
         + "  status string ,   "
         + "  type string )  "
         + "ROW FORMAT SERDE   "
         + "  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'  "
         + "STORED AS INPUTFORMAT   "
         + "  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'   "
         + "OUTPUTFORMAT   "
         + "  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat' "
         + "LOCATION\\n"
         + "  'cosn://<bucket_name>/<path>'  ",
         wholeTableName);
   statement.execute(createTableSql);
   // get meta data
   DatabaseMetaData metaData = connection.getMetaData();
   System.out.println("product = " + metaData.getDatabaseProductName());
   System.out.println("jdbc version = "
   + metaData.getDriverMajorVersion() + ", "
   + metaData.getDriverMinorVersion());
```

```
    ResultSet tables = metaData.getTables(null, dbName, tableName, null);
    while (tables.next()) {
        String name = tables.getString("TABLE_NAME");
        System.out.println("table: " + name);
        ResultSet columns = metaData.getColumns(null, dbName, name, null);
        while (columns.next()) {
        System.out.println(
        columns.getString("COLUMN_NAME") + "\\t" +
        columns.getString("TYPE_NAME") + "\\t" +
        columns.getInt("DATA_TYPE"));
    }
    columns.close();
    }
    tables.close();
    statement.close();
    connection.close();
}
}
```

数据查询

```
import java.sql.*;
public class DataTest {
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName("com.tencent.cloud.dlc.jdbc.DlcDriver");
            } catch (ClassNotFoundException e) {
    e.printStackTrace();
    return;
    }
    Connection connection = DriverManager.getConnection(
    "jdbc:dlc:<dlc_endpoint>?task_type=<task_type>&database_name=
<database_name>&datasource_connection_name=DataLakeCatalog&region=
<region>&data_engine_name=<data_engine_name>&result_type=<result_type>",
    "<secret_id>",
    "secret_key");
    Statement statement = connection.createStatement();
    String sql = "select * from dlc_test";
    statement.execute(sql);
    ResultSet rs = statement.getResultSet();
    while (rs.next()) {
        System.out.println(rs.getInt(1) + ":" + rs.getString(2));
        }
```

```
        rs.close();
        statement.close();
        connection.close();
        }
}
```

数据库客户端

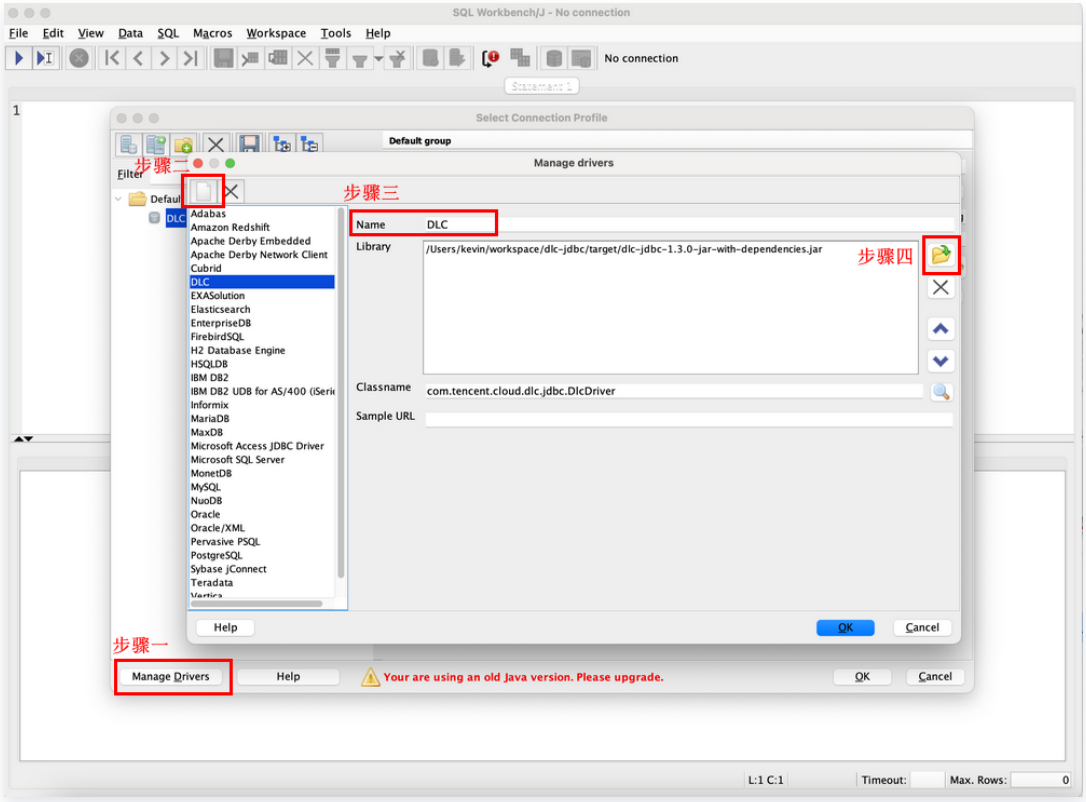
您可以将 DLC 的 JDBC 驱动包加载到 SQL 客户端，连接到 DLC 服务进行查询。

前置条件

1. 已开通数据湖计算 Data Lake Compute 服务。
2. 已下载上文中的 jdbc 驱动包。
3. 已下载并安装 SQL Workbench/J。

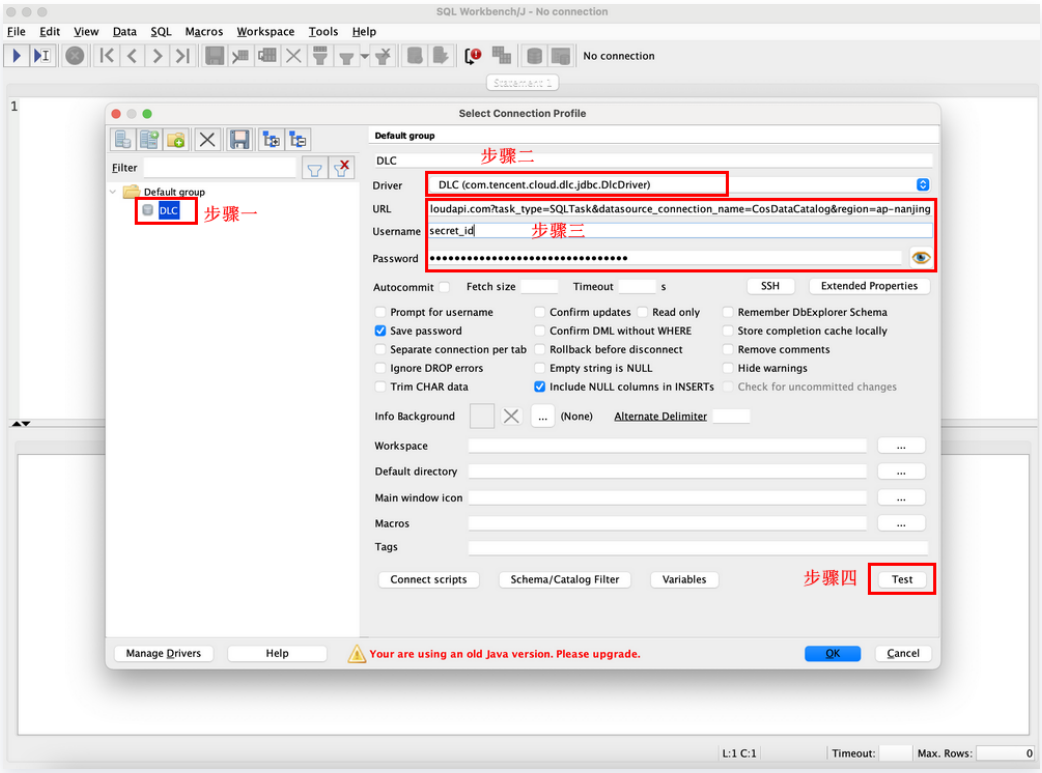
操作步骤

1. 通过 jdbc 驱动包创建 DLC Driver。

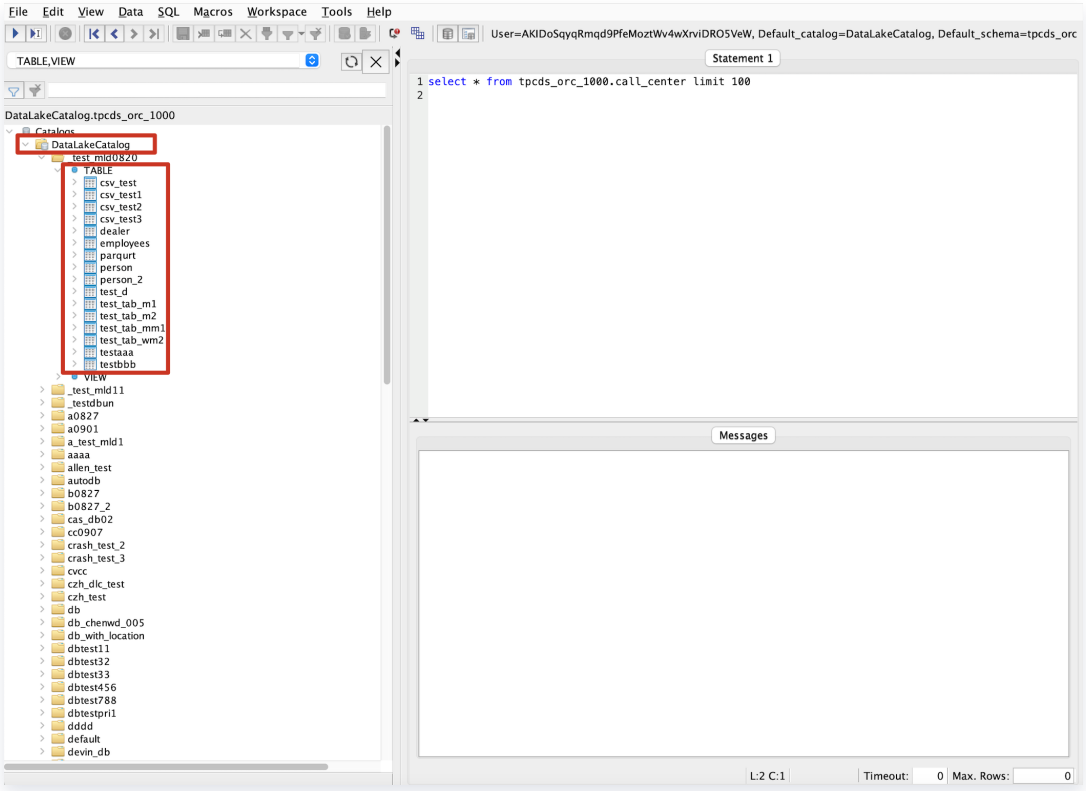


2. 连接 DLC，填入如下参数，单击 **test**，测试通过后，完成与 DLC 的连接。
   - Name：连接名称，用于标识与 DLC 的连接。
   - Username：对应于腾讯云用户的 secret_id。
   - Password：对应于腾讯云用户的 secret_key。

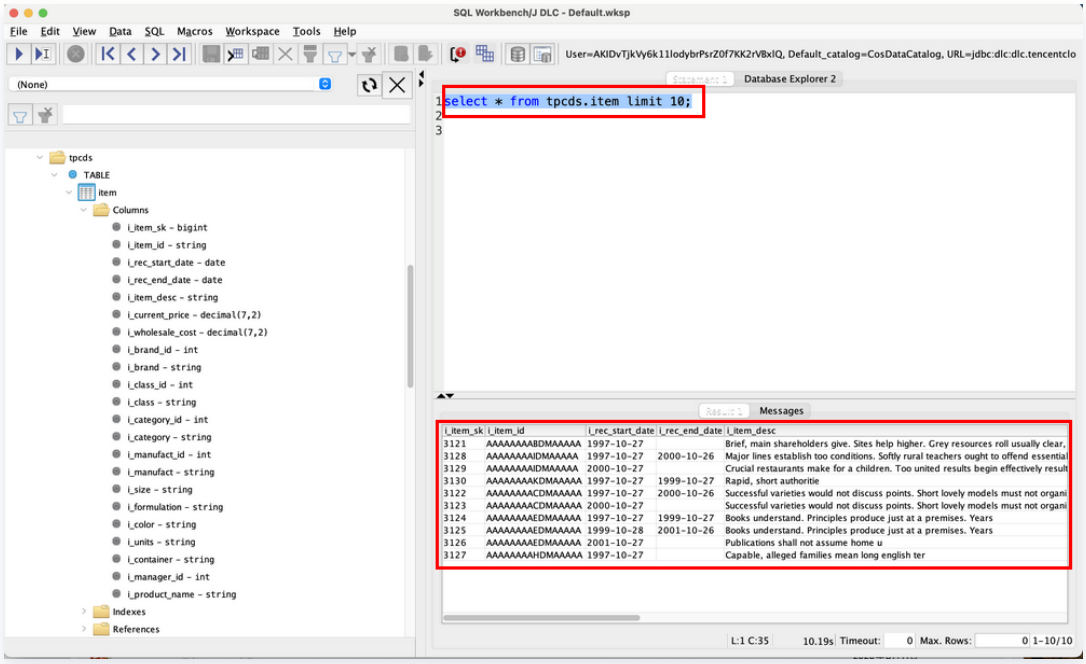
- URL：用于连接 DLC 的 URL，格式和上文中通过 jdbc 创建连接的 URL 一致。



3. 查看库表信息



4. 查询数据


SQL Workbench/J DLC - Default.wksp
File Edit View Data SQL Macros Workspace Tools Help
Database Explorer 2
select * from tpcds.item limit 10;
tpcds
TABLE
item
Columns
i_item_sk - bigint
i_item_id - string
i_rec_start_date - date
i_rec_end_date - date
i_item_desc - string
i_current_price - decimal(7,2)
i_wholesale_cost - decimal(7,2)
i_brand_id - int
i_brand - string
i_class_id - int
i_class - string
i_category_id - int
i_category - string
i_manufact_id - int
i_manufact - string
i_size - string
i_formulation - string
i_color - string
i_units - string
i_container - string
i_manager_id - int
i_product_name - string
Indexes
References
Messages
i_item_sk i_item_id i_rec_start_date i_rec_end_date i_item_desc
3121 AAAAAAAABDMAAAAA 1997-10-27 Brief, main shareholders give. Sites help higher. Grey resources roll usually clear,
3128 AAAAAAAAIDMAAAAA 1997-10-27 2000-10-26 Major lines establish too conditions. Softly rural teachers ought to offend essential
3129 AAAAAAAAIDMAAAAA 2000-10-27 Crucial restaurants make for a children. Too united results begin effectively result
3130 AAAAAAAAKDMAAAAA 1997-10-27 1999-10-27 Rapid, short authoritie
3122 AAAAAAAACDMAAAAA 1997-10-27 2000-10-26 Successful varieties would not discuss points. Short lovely models must not organi
3123 AAAAAAAACDMAAAAA 2000-10-27 Successful varieties would not discuss points. Short lovely models must not organi
3124 AAAAAAAAEDMAAAAA 1997-10-27 1999-10-27 Books understand. Principles produce just at a premises. Years
3125 AAAAAAAAEDMAAAAA 1999-10-28 2001-10-26 Books understand. Principles produce just at a premises. Years
3126 AAAAAAAAEDMAAAAA 2001-10-27 Publications shall not assume home u
3127 AAAAAAAAHDMAAAAA 1997-10-27 Capable, alleged families mean long english ter
L:1 C:35 10.19s Timeout: 0 Max. Rows: 0 1-10/10

Python 访问

最近更新时间：2023-12-13 21:41:32

数据湖计算 DLC 提供符合 DBAPI 2.0 标准的工具。您可以通过 Python 连接到 DLC 的 Presto/Spark 引擎，可以方便地使用 SQL 的方式操作 DLC 库表。

环境准备

1. Python 3.9及以上版本。
2. 安装 tencentcloud-dlc-connector。

```
pip install -i https://mirrors.tencent.com/pypi/simple/ tencentcloud-dlc-connector
```

使用示例

步骤一：连接到引擎

代码：

```
import tdlc_connector
import datetime
from tdlc_connector import constants

conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    token=None,
    endpoint=None,
    catalog=constants.Catalog.DATALAKECATALOG,
    engine="<ENGINE>",
    engine_type=constants.EngineType.AUTO,
    result_style=constants.ResultStyles.LIST,
    download=False,
    mode=constants.Mode.LASY,
    database='',
    config={},
    callback=None,
    callback_events=None,
    )
```

参数说明：

参数	说明
region	引擎所在地域，如 ap-nanjing，ap-beijing，ap-guangzhou，ap-shanghai，ap-chengdu，ap-chongqing，na-siliconvalley，ap-singapore, ap-hongkong
secret_id	腾讯云 SecretID
secret_key	腾讯云 SecretKey
token	（可选）临时密钥 Token
endpoint	（可选）连接服务节点
engine	使用的引擎名称，例如 “test_python”
engine_type	（可选）引擎类型：对应引擎名称的引擎类型， 默认值 constants.EngineType.AUTO 例如：AUTO、PRESTO、SPARK、SPARK_BATCH
result_style	（可选）返回结果的格式，可选 LIST/DICT
download	（可选）是否直接下载数据 True/False，详见 下载模式说明
mode	（可选）模式。支持 ALL/LASY/STREAM
database	（可选）默认数据库
config	（可选）提交到集群配置
callback	（可选）回调函数，函数签名 def cb(statement_id, status)
callback_events	（可选）回调触发事件，同callback配合使用，详见回调机制说明
driver_size	（可选）Driver 节点大小，默认值 constants.PodSize.SMALL （仅 SPARK_BATCH 集群有效） 可选值：SMALL、MEDIUM、LARGE、XLARGE、M_SMALL、M_MEDIUM、M_LARGE、M_XLARGE
executor_size	（可选）Executor 节点大小，默认值 constants.PodSize.SMALL（仅 SPARK_BATCH 集群有效） 可选值：SMALL、MEDIUM、LARGE、XLARGE、M_SMALL、M_MEDIUM、M_LARGE、M_XLARGE
executor_num	（可选）Executor 节点数量， 默认值1（仅SPARK_BATCH 集群有效）

executor_max_num	（可选）Executor 最大节点数量，若不等于 executor_num， 则开启资源动态分配（仅 SPARK_BATCH 集群有效）

下载模式说明：

序号	download	mode	说明
1	False	ALL	从接口获取所有数据，获取完成后才能 fetch 到数据
2	False	LASY	从接口获取数据，根据 fetch 的数据量延迟到服务端获取数据
3	False	STREAM	同 LASY 模式
4	True	ALL	从 COS 下载所有结果（需具备 COS 读权限）占用本地临时存储，数据量较大时推荐使用
5	True	LASY	从 COS 下载结果（需具备 COS 读权限），根据 fetch 数据量延迟下载文件
6	True	STREAM	实时的从 COS 读取结果流（需具备 COS 读权限），性能较慢，本地内存磁盘占用率极低

步骤二：执行 SQL

代码：

```
# 基本操作

cursor = conn.cursor()
count = cursor.execute("SELECT 1")
print(cursor.fetchone())                # 读取一行数据
for row in cursor.fetchall():           # 读取剩余多行数据
    print(row)

# 使用参数 pyformat 格式

cursor.execute("SELECT * FROM dummy WHERE date < %s",
datetime.datetime.now())
```

```
cursor.execute("SELECT * FROM dummy WHERE status in %s", (('SUCCESS', 'INIT',
'FAIL'),))
cursor.execute("SELECT * FROM dummy WHERE date < %(date)s AND status = %
(status)s", {'date': datetime.datetime.now(), 'status': 'SUCCESS'})

# 使用BULK方式

cursor.executemany("INSERT INTO dummy VALUES(%s, %s)", [('张三', 18), ('李四', 20)])
```

基本操作流程

上述代码流程如下：

1. 通过 `conn.cursor()` 创建了一个游标对象。
2. 通过 `cursor.execute("SELECT 1")` 执行了一条 SQL 查询语句，并将结果赋值给变量 `count`。
3. 通过调用 `cursor.fetchone()` 方法读取了一行数据，并将其打印出来。
4. 在一个循环中调用了 `cursor.fetchall()` 方法来读取剩余的多行数据，并逐行打印出来。

参数传递方式

支持两种不同的参数传递方式：

- 使用 pyformat 格式：在执行 SQL 语句时，可以使用%s作为占位符，并将实际参数以元组或字典形式传入。
- 使用 BULK 方式：通过调用 `executemany()` 方法可以批量插入多个记录到数据库表中。

特性功能

回调机制使用说明

```
import tdlc_connector
import datetime
from tdlc_connector import constants


def tdlc_connector_callback(statement_id, state):
    '''
    parmas: statement_id 任务id
    params: state 任务状态，枚举值 参考constants.TaskStatus
    '''
    print(statement_id, state)


conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
```

```
    engine="<ENGINE>",
    engine_type=constants.EngineType.SPARK,
    result_style=constants.ResultStyles.LIST,
    callback=tdlc_connector_callback,
    callback_events=[constants.CallbackEvent.ON_INIT,
constants.CallbackEvent.ON_SUCCESS]
    )

cursor = conn.cursor()
cursor.execute("SELECT 1")
cursor.fetchone()

# callback 函数会在任务初始化 和 任务成功时调用
```

提交任务到作业集群

当前已支持提交任务到 Spark 作业集群，具体请参考如下示例。

```
from tdlc_connector import constants

conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    engine="<ENGINE>",                    # 请选择 spark 作业引擎
    result_style=constants.ResultStyles.LIST,
    driver_size=constants.PodSize.SMALL,        # 选择 Driver 规格
    executor_size=constants.PodSize.SMALL,      # 选择 Executor 规格
    executor_num=1,                       # 设置 Executor 数量
    executor_max_num=1,                   # 设置 Executor 最大数量， 如果 不等于
{executor_num}，则开启动态资源分配
)
```

> **说明：**
> 如需使用该特性，请将 connector 升级至 >= 1.1.0。

自动推断引擎类型

当前指定使用引擎后无需再指定引擎类型，connector 会自动推断，具体请参考如下示例。

```
from tdlc_connector import constants

conn = tdlc_connector.connect(region="<REGION>",
```

```
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    engine="<ENGINE>",
    engine_type=constants.EngineType.AUTO      # 可设置成AUTO 或者 不传入该参数，驱
动自动推断
)
```

说明：
如需使用该特性，请将 connector 升级至 >= 1.1.0。

空值转换

当前结果集基于CSV格式存储，引擎默认将空值转换成空字符串，如果需要区分空值，请指定空值表示符, 例如“\1”，引擎查询结果会将空值转换成 “\1”，同时驱动会将 “\1”字段转换成 None，具体请参考如下示例。

```
from tdlc_connector import constants, formats

formats.FORMAT_STRING_NULL = '\1'

conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    engine="<ENGINE>",
    result_style=constants.ResultStyles.LIST
)
```

说明：
空值转换目前仅支持 SparkSQL集群，请将 connector 升级至 >= 1.1.3。