ALIBABA CLOUD

Alibaba Cloud

大数据计算服务

JDBC参考

文档版本：20220330

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.版本更新记录 ---------- 05
2.概述 ---------- 06
3.使用说明 ---------- 08

1.版本更新记录

本文为您介绍JDBC近期版本的更新说明，基于此您可以了解JDBC对应版本中的新增功能、增强功能内容。

JDBC近期版本的更新说明如下，详细信息请单击对应版本链接获取。

版本	变更类型	描述
v3.2.9	新功能	增加可选连接串参数 `useProjectTimeZone` 。该参数默认值为False。配置 `useProjectTimeZone=true` 时， `ResultSet#getDate` 、 `ResultSet#getTime` 和 `ResultSet#getTimestamp` 将会按当前项目所在时区构建返回值。
	增强功能	优化执行失败时的日志。
	缺陷修复	禁止Maven AspectJ插件，避免潜在的性能损耗。
v3.2.8	新功能	支持查询加速（MCQA）功能。
	增强功能	• 在查询加速模式下，添加选项 `instanceTunnelMaxRecord` 以设置读取结果的数据行数，默认为-1，表示不限制，仅在JDBC 3.2.7及以上版本才支持。 • 在查询加速模式下，添加选项 `instanceTunnelMaxSize` 以设置读取结果的数据大小，单位为Bytes，默认为-1，表示不限制，仅在JDBC 3.2.7及以上版本才支持。 • 在查询加速模式下，添加选项 `disableConnectionSetting` 以禁用连接级别设置。 • 在查询加速模式下，添加选项 `alwaysFallback` ，设置为True时，表示在以上几种场景下全部回退到离线模式，仅在JDBC 3.2.3及以上版本才支持。

2.概述

MaxCompute JDBC驱动是MaxCompute提供的Java数据库连接（Java Database Connectivity）接口。您可以通过标准的JDBC接口基于MaxCompute执行海量数据的分布式计算查询。MaxCompute JDBC驱动还可以用于连接MaxCompute和支持JDBC的工具。

JDBC使用说明

- 使用JDBC连接MaxCompute。详情请参见使用说明。
- 使用JDBC连接商业智能（BI）分析工具，对MaxCompute中的数据进行可视化分析，相关文档如下：
  - Tableau连接MaxCompute
  - Davinci连接MaxCompute
  - Yonghong BI连接MaxCompute
- 使用JDBC连接数据库管理工具，管理MaxCompute项目，相关文档如下：
  - DBeaver连接MaxCompute
  - DataGrip连接MaxCompute
  - SQL Workbench/J连接MaxCompute

注意事项

- 通过MaxCompute JDBC驱动执行SQL并获取结果，需要执行账号满足以下要求：
  - 是项目空间的成员。
  - 有项目空间的CreateInstance权限。
  - 有目标表的Select和Download权限。

    > **说明**
    > - 1.9及之前版本的MaxCompute JDBC驱动对每个查询都会创建临时表，并通过Tunnel从临时表获取结果。您使用这些版本的JDBC需要具备CreateTable权限。
    > - 2.2及之后版本的MaxCompute JDBC驱动不再创建临时表，直接通过Instance Tunnel获取查询结果，没有CreateTable权限要求限制。
    >
    > MaxCompute权限详情请参见权限列表。

- MaxCompute提供了数据保护功能。当数据保护模式开启时，您无法将数据转移到项目空间之外。2.4之前版本的JDBC无法获取 `result set` 。2.4及之后版本的JDBC可以获得不超过READ_TABLE_MAX_ROW所定义行数的数据，详情请参见项目空间操作。数据保护功能详情请参见数据保护机制。
- MaxCompute 2.0数据类型版本支持较多数据类型，例如TINYINT、SMALLINT、DATETIME、TIMESTAMP、ARRAY、MAP和STRUCT等。您如果需要使用这些新类型，在执行SQL之前需要执行以下语句，打开MaxCompute 2.0数据类型开关。详情请参见数据类型版本说明。

```
set odps.sql.type.system.odps2=true
```

常见问题

- 如何查看MaxCompute JDBC驱动的日志？
  - MaxCompute JDBC驱动的日志默认放在驱动JAR包所在的同级目录下，文件名为jdbc.log。
  - 如果代码与驱动打包为一个Uber JAR，日志将会在Uber JAR所在的同级目录。

MaxCompute JDBC驱动的日志记录了调用JDBC接口的详细信息，包括调用的类名、方法名、行数、参数和返回值等。您可以使用这些信息进行调试。

- 如何获取MaxCompute Logview URL?

  MaxCompute JDBC驱动是基于MaxCompute Java SDK的封装。与MaxCompute客户端、MaxCompute Studio以及DataWorks一样，您通过MaxCompute JDBC驱动执行SQL时，会生成Logview URL。您可以通过Logview查看作业执行状态、追踪作业进度和获取作业执行结果。Logview URL可以通过properties.log4j参数配置，默认打印到StdErr。

- 是否支持连接池与Auto-Commit?

  MaxCompute提供Rest服务，和传统数据库维持长连接的方式不同。MaxCompute JDBC驱动创建Connection是十分轻量的操作，无须针对MaxCompute JDBC刻意使用连接池。MaxCompute JDBC也支持使用连接池的场景。

  MaxCompute不支持Transaction，每个查询动作都会即时体现到服务端，即默认为Auto-Commit的行为。MaxCompute JDBC驱动不支持关闭Auto-Commit模式。

- 如何获取分区字段和类型的描述?

  您可以通过 `Connection.getMetadata()` 方法获取DatabaseMetaData对象，通过DatabaseMetaData的 `getColumns()` 方法获取所有列的元数据。

3.使用说明

本文为您介绍如何下载JDBC和连接MaxCompute，并提供示例代码。

JDBC下载

您可以通过GitHub或Maven库获取MaxCompute各版本的JAR包。推荐您下载包含完整依赖jar-with-dependencies的JAR包。

通过Maven方式使用MaxCompute JDBC的项目对象模型POM（Project Object Model）的示例如下。

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-jdbc</artifactId>
  <version>3.0.1</version>
  <classifier>jar-with-dependencies</classifier>
</dependency>
```

> **说明** MaxCompute JDBC驱动是开源代码项目，项目地址为aliyun-odps-jdbc。
>
> MaxCompute欢迎您参与JDBC驱动的开发和改进工作。您可以在该项目的Issues页面反馈问题，或通过Pull requests页面对源代码进行改进。使用Issues及Pull requests时，请您遵循开源项目的模板要求。

连接MaxCompute

1. 加载MaxCompute JDBC驱动。

   ```
   Class.forName("com.aliyun.odps.jdbc.OdpsDriver");
   ```

2. 通过DriverManager创建Connection。

   ```
   Connection cnct = DriverManager.getConnection(url, accessId, accessKey);
   ```

   - url：格式为 `jdbc:odps:<maxcompute_endpoint>?project=<maxcompute_project_name>[&useProjectTimeZone={true|false}]` 。其中：
     - <maxcompute_endpoint>：MaxCompute服务所在区域的Endpoint。例如，华东1（杭州）区域的外网Endpoint为 `http://service.cn-hangzhou.maxcompute.aliyun.com/api` 。Endpoint的配置信息详情请参见Endpoint。
     - <maxcompute_project_name>：MaxCompute项目空间名称。
     - useProjectTimeZone：是否使用MaxCompute项目空间的时区。

     命令示例如下。

     ```
     jdbc:odps:http://service.cn-hangzhou.maxcompute.aliyun.com/api?project=test_project&useProjectTimeZone=true;
     ```

   - accessId：创建项目空间的AccessKey ID。
   - accessKey：创建项目空间的AccessKey ID对应的AccessKey Secret。

> **说明** AccessKey ID和AccessKey Secret的创建和查看，请参见准备阿里云账号。

3. 执行查询。

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

示例代码

- 删除表、创建表和获取Metadata

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class Main {
    private static final String DRIVER_NAME = "com.aliyun.odps.jdbc.OdpsDriver";
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName(DRIVER_NAME);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
        Connection conn = DriverManager.getConnection(
            "jdbc:odps:<maxcompute_endpoint>?project=<maxcompute_project>",
            "aliyun accessId", "aliyun accessKey");
        // create a table
        Statement stmt = conn.createStatement();
        final String tableName = "jdbc_test";
        stmt.execute("DROP TABLE IF EXISTS " + tableName);
        stmt.execute("CREATE TABLE " + tableName + " (key BIGINT, value STRING)");
        // get meta data
        DatabaseMetaData metaData = conn.getMetaData();
        System.out.println("product = " + metaData.getDatabaseProductName());
        System.out.println("jdbc version = "
            + metaData.getDriverMajorVersion() + ", "
            + metaData.getDriverMinorVersion());
        ResultSet tables = metaData.getTables(null, null, tableName, null);
        while (tables.next()) {
            String name = tables.getString("TABLE_NAME");
            System.out.println("inspecting table: " + name);
            ResultSet columns = metaData.getColumns(null, null, name, null);
            while (columns.next()) {
                System.out.println(
                    columns.getString("COLUMN_NAME") + "\t" +
                        columns.getString("TYPE_NAME") + "(" +
                        columns.getInt("DATA_TYPE") + ")");
            }
            columns.close();
        }
        tables.close();
        stmt.close();
        conn.close();
    }
}
```

返回结果示例如下。

```
product = MaxCompute/ODPS
jdbc version = 3, 0
inspecting table: jdbc_test
key    BIGINT(-5)
value    STRING(12)
```

- 更新表

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Main {
    private static final String DRIVER_NAME = "com.aliyun.odps.jdbc.OdpsDriver";
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName(DRIVER_NAME);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
        Connection conn = DriverManager.getConnection(
            "jdbc:odps:<maxcompute_endpoint>?project=<maxcompute_project>",
            "aliyun accessId", "aliyun accessKey");
        Statement stmt = conn.createStatement();
        // The following DML also works
        //String dml = "INSERT INTO jdbc_test SELECT 1, \"foo\"";
        String dml = "INSERT INTO jdbc_test VALUES(1, \"foo\")";
        int ret = stmt.executeUpdate(dml);
        assert ret == 1;
        stmt.close();
        conn.close();
    }
}
```

- 批量更新表

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class Main {
    private static final String DRIVER_NAME = "com.aliyun.odps.jdbc.OdpsDriver";
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName(DRIVER_NAME);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
        Connection conn = DriverManager.getConnection(
            "jdbc:odps:<maxcompute endpoint>?project=<maxcompute project>",
            "aliyun accessId", "aliyun accessKey");
        PreparedStatement pstmt = conn.prepareStatement("INSERT INTO jdbc_test VALUES(?,
?)");
        pstmt.setLong(1, 1L);
        pstmt.setString(2, "foo");
        pstmt.addBatch();
        pstmt.setLong(1, 2L);
        pstmt.setString(2, "bar");
        pstmt.addBatch();
        int[] ret = pstmt.executeBatch();
        assert ret[0] == 1;
        assert ret[1] == 1;
        pstmt.close();
        conn.close();
    }
}
```

- 查询表

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class Main {
    private static final String DRIVER_NAME = "com.aliyun.odps.jdbc.OdpsDriver";
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName(DRIVER_NAME);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
        Connection conn = DriverManager.getConnection(
            "jdbc:odps:<maxcompute endpoint>?project=<maxcompute project>",
            "aliyun accessId", "aliyun accessKey");
        ResultSet rs;
        Statement stmt = conn.createStatement();
        String sql = "SELECT * FROM JDBC_TEST";
        stmt.executeQuery(sql);
        ResultSet rset = stmt.getResultSet();
        while (rset.next()) {
            System.out.println(String.valueOf(rset.getInt(1)) + "\t" + rset.getString(2))
;
        }
    }
}
```