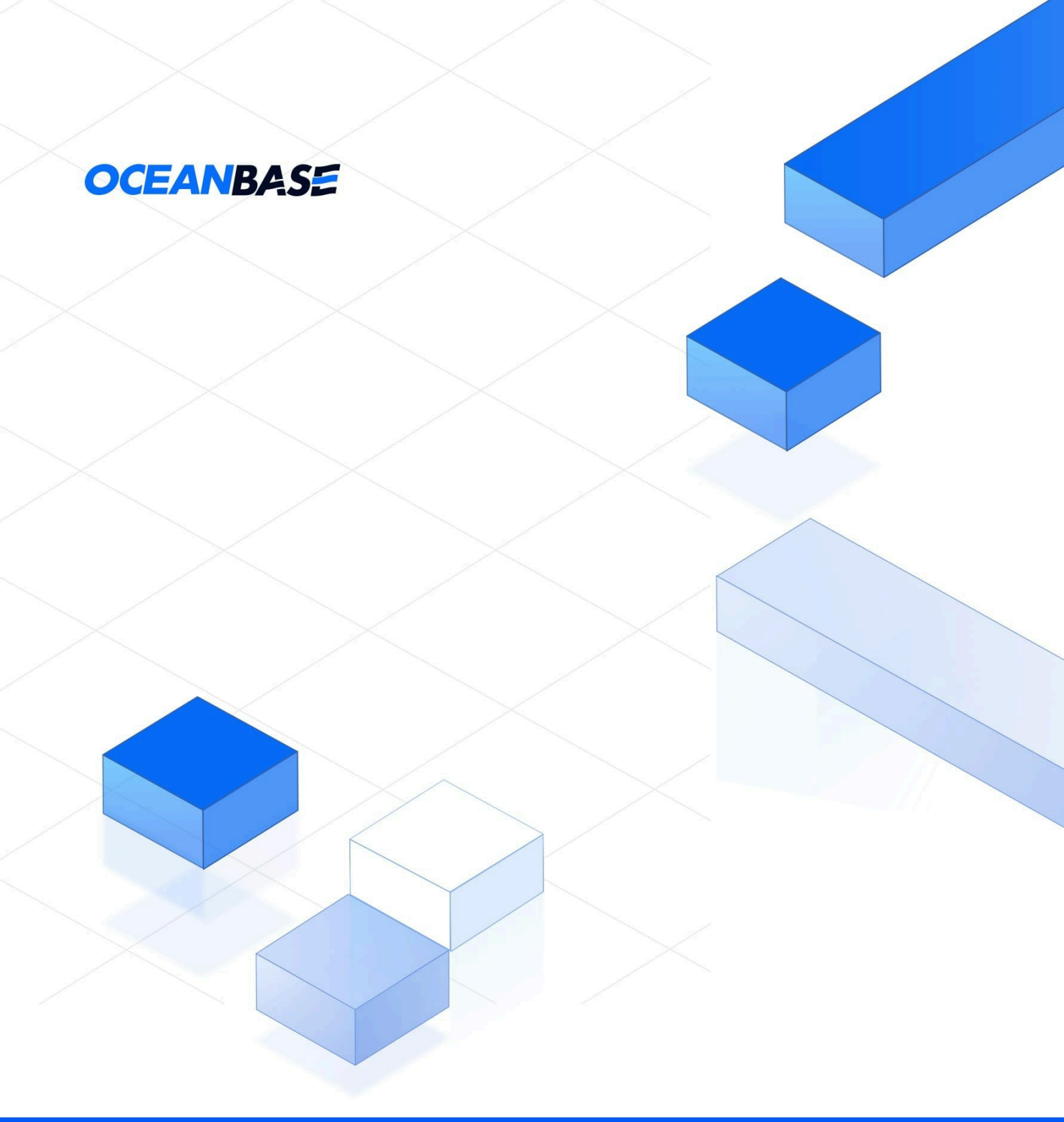

OceanBase 数据库

参考指南--OceanBase 数据库插件

| 产品版本：V4.5.0 | 文档版本：20251219

声明

北京奥星贝斯科技有限公司版权所有©2024，并保留一切权利。

未经北京奥星贝斯科技有限公司事先书面许可，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。

商标声明

OCEANBASE及其他OceanBase相关的商标均为北京奥星贝斯科技有限公司所有。本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

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。北京奥星贝斯科技有限公司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，并在北京奥星贝斯科技有限公司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北京奥星贝斯科技有限公司授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

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 OceanBase 数据库插件概述

从 OceanBase 数据库 V4.3.5 BP1 版本开始，OceanBase 引入了插件机制，旨在为用户提供一种轻量、灵活且高效的功能扩展方式。通过插件机制，用户可以根据自身需求定制和增强 OceanBase 数据库的功能。

1.0.0.1 注意

OceanBase 数据库插件机制目前为实验特性，暂不建议在生产环境中使用。

1.1 如何开发和使用插件

OceanBase 数据库支持使用 C/C++ 语言开发插件，并提供了一套完整的插件开发包（Plugin Development Kit, PDK）。只要您的开发环境中已安装 C/C++ 开发工具链，并配置了 OceanBase 插件开发包，即可快速上手插件开发。

1.2 支持场景

当前，OceanBase 数据库插件机制主要支持以下场景：

- 分词器插件：用于扩展数据库的全文搜索能力，允许用户自定义分词逻辑，满足多样化的文本处理需求。
- 外表 JDBC 插件：通过外表 JDBC 插件，可以访问 JDBC 支持的数据源。当前版本仅支持 MySQL 模式。

1.2.0.1 说明

从 V4.4.1 版本开始支持外表 JDBC 插件服务。同时 OceanBase 数据库也基于 JDBC 插件服务发布了 MySQL 外表插件。

1.3 相关文档

- 插件机制
- 分词器插件
- MySQL 外表插件

2 分词器插件

本文将介绍如何在 OceanBase 数据库安装和使用分词器插件。分词器插件可用于全文索引功能，支持自定义分词逻辑以满足特定业务需求。

2.0.0.1 注意

当前分词器插件为实验特性，暂不建议在生产环境使用。

2.1 前提条件

在安装和使用分词器插件之前，请确保以下条件已满足：

- 操作系统支持 yum 包管理工具。

2.1.0.1 说明

如果您使用的系统不支持 `yum`，可以通过其他方式安装上述依赖项，例如手动下载 RPM 包或使用其他包管理工具。

- 已部署正常运行的 OceanBase 数据库集群。
- 具备系统租户权限，用于修改配置参数和重启集群。

2.2 操作步骤

2.2.1 步骤一：安装开发环境

分词器插件的开发需要 C/C++ 编译环境以及 OceanBase 提供的插件开发工具包。执行以下命令安装 C/C++ 开发环境和 OceanBase 插件开发包：

1. 安装基础编译工具。

   安装了编译 C/C++ 程序所需的基本工具和库。

   ```
   yum install -y cmake make glibc-devel glibc-headers gcc gcc-c++
   ```

2. 配置 OceanBase 软件源。

   添加 OceanBase 的官方软件源，以便后续通过 yum 安装 OceanBase 相关的工具和依赖。

   a. 安装 yum-utils 工具。

      ```
      yum install -y yum-utils
      ```

   b. 添加 OceanBase 软件源。

```
yum-config-manager --add-repo https://mirrors.aliyun.com/oceanbase
/OceanBase.repo
```

3. 安装插件开发套件。

```
yum install -y oceanbase-plugin-dev-kit
```

安装完成后，可以在 `/usr/share/examples/ObPlugin/ftparser` 目录下看到 OceanBase 分词器插件样例代码文件 `space_ftparser.cpp` 。

```
ls /usr/share/examples/ObPlugin/ftparser
```

返回结果如下：

```
CMakeLists.txt space_ftparser.cpp
```

2.2.2 步骤二：获取开发模板

复制 OceanBase 分词器插件样例代码 `space_ftparser.cpp` 到自己的开发目录下，即可以修改 `space_ftparser.cpp` 文件开发自己的分词器插件。

示例如下：

```
[root@xxx packages]# cp /usr/share/examples/ObPlugin/ftparser/* /home/admin
/test_plugin_dev
```

```
[root@xxx packages]# ls /home/admin/test_plugin_dev
```

返回结果如下：

```
CMakeLists.txt space_ftparser.cpp
```

样例代码的核心文件是 `space_ftparser.cpp` ，您可以根据业务需求修改该文件以实现自定义的分词逻辑。

2.2.3 步骤三：编译和安装插件

1. 修改构建配置（ `CMakeLists.txt` ）。

在样例代码的根目录下，您会找到一个 `CMakeLists.txt` 文件。该文件中使用 "TODO" 标记了需要修改的内容，包括以下部分：

- `PLUGIN_NAME`：当前插件的名称，也是项目和生成的动态链接库的名称。请将其修改为您需要的名称。
- `SOURCES`：实现文件列表，可以包含 C 或 C++ 源文件。当您新增实现文件时，请在此处添加文件路径。注意不要将头文件列入此列表。

2. 执行编译。

   执行下面的步骤进行编译：

   a. 切换到工作目录。

   ```
   cd /your/work/path/ftparser
   ```

   切换到您的分词器插件开发目录。请将 `/your/work/path/ftparser` 替换为实际的开发目录路径。

   b. 创建构建目录。

   ```
   mkdir -p build
   ```

   创建一个名为 `build` 的目录，用于存放编译过程中生成的中间文件和最终产物。`-p` 参数确保即使目录已存在也不会报错。

   c. 进入构建目录。

   ```
   cd build
   ```

   进入刚刚创建的 `build` 目录。后续的编译操作将在该目录下进行，以保持源码目录的整洁。

   d. 配置构建环境。

   ```
   cmake ..
   ```

   运行 `cmake` 命令，读取上一级目录中的 `CMakeLists.txt` 文件，生成编译所需的 `Makefile` 文件。`..` 表示 `CMakeLists.txt` 文件位于上一级目录中。

   e. 编译源码。

   ```
   make
   ```

运行 make 命令，根据生成的 Makefile 文件编译源码，生成动态链接库文件（例如 libexample_ftparser.so ）。

编译成功后，会在当前目录（即 build 目录）下生成动态链接库文件。

3. 复制编译产物。

   编译完成后，会在 build 目录下生成动态链接库文件（例如 libexample_ftparser.so ）。将该文件复制到 OceanBase 集群中每个 Observer 节点的 plugin_dir 目录下。

   ```
   cp libexample_ftparser.so /path/to/plugin_dir/
   ```

   将生成的动态链接库文件复制到 OceanBase 的插件目录。请将 /path/to/plugin_dir/ 替换为实际的 plugin_dir 路径（可通过查询系统参数 plugin_dir 获取）。

4. 加载插件。

   使用系统租户登录 OceanBase 数据库，修改配置参数 plugins_load 以加载插件：

   ```
   ALTER SYSTEM SET plugins_load='libexample_ftparser.so';
   ```

5. 重启集群，使插件生效。

   - 使用 OceanBase 部署工具（OBD）管理的 OceanBase 集群，可以使用以下命令重启集群。

     ```
     obd cluster restart <cluster_name>
     ```

     请将 <cluster_name> 替换为实际的集群名称。

   - 使用 OceanBase 云平台（OCP）管理的集群，可以在 OCP 上直接重启集群。

6. 查看已安装的分词器插件。

   ```
   select * from oceanbase.GV$OB_PLUGINS;
   ```

   返回结果如下：

   ```
   +-----------+----------+-----------------+--------+----------+-----------------------
   +-----------------+-----------------+------------------+----------------------
   +---------------+-------------------------------------------+
   | SVR_IP | SVR_PORT | NAME | STATUS | TYPE | LIBRARY | LIBRARY_VERSION |
   ```

```
LIBRARY_REVISION | INTERFACE_VERSION | AUTHOR | LICENSE | DESCRIPTION |
+----------+---------+-----------------+--------+---------+-----------------------
+----------------+-----------------+------------------+----------------------
+--------------+-------------------------------------------+
| 127.0.0.1 | 55801 | ngram | READY | FTPARSER | NULL | 1.0.0 | NULL | 0.1.0 |
OceanBase Corporation | Mulan PubL v2 | This is a ngram fulltext parser plugin. |
| 127.0.0.1 | 55801 | beng | READY | FTPARSER | NULL | 1.0.0 | NULL | 0.1.0 |
OceanBase Corporation | Mulan PubL v2 | This is a basic english parser plugin. |
| 127.0.0.1 | 55801 | space | READY | FTPARSER | NULL | 1.0.0 | NULL | 0.1.0 |
OceanBase Corporation | Mulan PubL v2 | This is a default whitespace parser
plugin. |
| 127.0.0.1 | 55801 | example_ftparser | READY | FTPARSER | libexample_ftparser.so |
1.0.0 | NULL | 0.1.0 | OceanBase Corporation | Mulan PSL v2 | This is an example
ftparser. |
+----------+---------+-----------------+--------+---------+-----------------------
+----------------+-----------------+------------------+----------------------
+--------------+-------------------------------------------+
```

其中 `LIBRARY` 是 `NULL` 的表示是内置分词器。

2.2.4 步骤四：测试分词器插件

1. 创建表并使用 `WITH PARSER` 子句指定分词器 `example_ftparser` 。

```
CREATE TABLE t_example(
c1 INT,
c2 VARCHAR(200),
c3 TEXT,
FULLTEXT INDEX (c2, c3) WITH PARSER example_ftparser
);
```

2. 向表中插入测试数据。

```
INSERT INTO t_example (c1, c2, c3) VALUES
(1, 'Alice', 'Alice loves programming and enjoys long walks.'),
(2, 'Bob', 'Bob is an avid reader and a coffee enthusiast.'),
(3, 'Charlie', 'Charlie is a skilled musician who plays the guitar.'),
(4, 'Diana', 'Diana is passionate about painting and arts.'),
(5, 'Eve', 'Eve is a fitness coach and a healthy lifestyle advocate.');
```

3. 查询包含关键词 `loves` 的记录。

```
SELECT * FROM t_example WHERE MATCH(c2, c3) AGAINST ('loves') > 0;
```

返回结果如下：

```
+------+-------+-----------------------------------------------+
| c1 | c2 | c3 |
+------+-------+-----------------------------------------------+
| 1 | Alice | Alice loves programming and enjoys long walks. |
+------+-------+-----------------------------------------------+
1 row in set
```

4. 查询包含关键词 `reader` 的记录。

```
SELECT * FROM t_example WHERE MATCH(c2, c3) AGAINST ('reader') > 0;
```

返回结果如下：

```
+------+------+-------------------------------------------------+
| c1 | c2 | c3 |
+------+------+-------------------------------------------------+
| 2 | Bob | Bob is an avid reader and a coffee enthusiast. |
+------+------+-------------------------------------------------+
1 row in set
```

5. 测试分词得分。

```

```

```
SELECT c1,
MATCH (c2, c3) AGAINST ('he loves programming and reading') AS score,
c2,
c3
FROM t_example;
```

返回结果如下：

```
+------+-------------------+---------
+-------------------------------------------------------------+
| c1 | score | c2 | c3 |
+------+-------------------+---------
+-------------------------------------------------------------+
| 1 | 2.665294094128556 | Alice | Alice loves programming and enjoys long walks. |
| 2 | 0.2849740932642488 | Bob | Bob is an avid reader and a coffee enthusiast. |
| 3 | 0 | Charlie | Charlie is a skilled musician who plays the guitar. |
| 4 | 0.2989130434782609 | Diana | Diana is passionate about painting and arts. |
| 5 | 0.2722772277227723 | Eve | Eve is a fitness coach and a healthy lifestyle
advocate. |
+------+-------------------+---------
+-------------------------------------------------------------+
5 rows in set
```

3 MySQL 外表插件

本文将介绍如何在 OceanBase 数据库安装和使用外表 JDBC 插件。通过外表 JDBC 插件，可以访问 JDBC 支持的数据源（当前仅支持访问 MySQL 数据库数据源）。

注意

- 从 V4.4.1 版本开始支持外表 JDBC 插件服务。同时 OceanBase 数据库也基于 JDBC 插件服务发布了 MySQL 外表插件。
- 当前该功能为实验特性，暂不建议在生产环境使用。

3.1 功能限制

- OceanBase 数据库 Oracle 模式：当前该功能不支持 Oracle 模式。
- 数据类型：`Array` 类型暂未支持。
- 查询限制：
  - 不支持并发查询。
  - 同一数据源多表 `JOIN` 不会作为一条 SQL 下推到数据源。
  - 聚合函数、`LIMIT` 等不会下推。
- 依赖管理：jar 包不支持动态加载，在进程启动前就需要把 jar 包放在指定的目录。
- 超时设置：无法通过插件配置查询超时时间。
- 参数修改：表 `PARAMETERS` 属性不支持修改，必须重新建表。

3.2 前提条件

- 已完成部署 OceanBase 数据库并且创建了 MySQL 模式用户租户。
- 已安装 JDK（版本 ≥ 11），并配置好 `JAVA_HOME` 环境变量。

3.3 配置 MySQL 外表插件的 jar 包路径

从 MySQL 外表插件下载地址 下载 MySQL 外表插件的 jar 包，然后把 MySQL 外表插件的 jar 包放到指定目录（配置项 `ob_java_connector_path` 目录下）。

示例如下：

1. 创建存放 jar 包放到指定目录。

```
mkdir -p /jdbc/plugin/jar/package/directory
```

2. 进入到目录。

```
cd /jdbc/plugin/jar/package/directory
```

3. 下载 MySQL 外表插件的 jar 包。

```
wget https://github.com/oceanbase/oceanbase-plugins/releases/download
/external-v1.0.0/oceanbase-external-plugin-1.0.0-jar-with-dependencies.jar
```

3.4 配置 OceanBase 数据库参数（重启后生效）

说明

如果是初次使用 JAVA SDK 环境，不需要重启 observer 进程。

1. 启用 Java。

   示例如下：

   ```
   ALTER SYSTEM SET ob_enable_java_env = true;
   ```

   该设置的详细介绍信息，参见 ob_enable_java_env。

2. 设置当前 OBServer 运行节点上的 java home 目录。

   # 说明

   该路径来自于 JDK 的 HOME 目录，设置为环境中的 `$JAVA_HOME` 即可。

   示例如下：

   ```
   ALTER SYSTEM SET ob_java_home = "/java/home/path";
   ```

   该设置的详细介绍信息，参见 ob_java_home。

3. 设置 JDBC 插件 jar 包目录。

   # 说明

   只需要设置 MySQL 外表插件 jar 包的目录，不需要设置 jar 包名称，会自动展开目录下的 jar 包。

   示例如下：

```
ALTER SYSTEM SET ob_java_connector_path = "/jdbc/plugin/jar/package
/directory";
```

该设置的详细介绍信息，参见 ob_java_connector_path。

4. 设置 java 环境启动的相关配置项
   1. 创建对应的日志文件夹路径。

      ```
      mkdir -p /home/user/jvmlogs
      mkdir -p /home/user/jvmlogs/heapdumps
      ```

   2. 设置 java 运行的 jvm 启动配置项。

      示例如下：

      ```
      ALTER SYSTEM SET ob_java_opts = "-Djdk.lang.
      processReaperUseDefaultStackSize=true -XX:
      +HeapDumpOnOutOfMemoryError -Xmx2048m -Xms2048m -Xloggc:/home
      /user/jvmlogs/gc.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:
      +HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/home/user/jvmlogs
      /heapdumps/ -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:
      GCLogFileSize=100M -XX:+UseG1GC -XX:-CriticalJNINatives --add-opens=java.
      base/java.nio=org.apache.arrow.memory.core,ALL-UNNAMED -Darrow.
      allocation.manager.type=Netty";
      ```

      该设置的详细介绍信息，参见 ob_java_opts。

      # 注意

      - 该配置项变更需要重启 observer 进程，由于当前使用的内存拷贝为数据流直接拷贝到 cpp 内存堆上，可以适当减少 `-Xmx2048m -Xms2048m` 的设置。
      - 相关 gc 日志文件，需要对应的配置文件夹路径存在即可。如果不存在对应路径，则相关的日志文件不存在。

5. 重启 OceanBase 数据库，检查插件是否安装成功。

   示例如下：

```
SELECT *
FROM oceanbase.GV$OB_PLUGINS
WHERE TYPE = 'EXTERNAL TABLE'
AND STATUS = 'READY';
```

返回结果如下：

```
+--------------+----------+-------+--------+---------------+---------
+-----------------+-----------------+------------------+----------------------
+--------------+-----------------------------------------------------+
| SVR_IP | SVR_PORT | NAME | STATUS | TYPE | LIBRARY | LIBRARY_VERSION |
LIBRARY_REVISION | INTERFACE_VERSION | AUTHOR | LICENSE | DESCRIPTION |
+--------------+----------+-------+--------+---------------+---------
+-----------------+-----------------+------------------+----------------------
+--------------+-----------------------------------------------------+
| 127.0.0.1 | 55803 | java | READY | EXTERNAL TABLE | NULL | 0.1.0 | NULL | 0.2.0 |
OceanBase Corporation | Mulan PubL v2 | This is the java external table data
source plugin. |
| 127.0.0.1 | 55803 | jdbc | READY | EXTERNAL TABLE | NULL | 0.1.0 | NULL | 0.2.0 |
OceanBase Corporation | Mulan PubL v2 | This is a java external data source |
| 127.0.0.1 | 55803 | mysql | READY | EXTERNAL TABLE | NULL | 0.1.0 | NULL | 0.2.0 |
OceanBase Corporation | Mulan PubL v2 | This is a java external data source |
+--------------+----------+-------+--------+---------------+---------
+-----------------+-----------------+------------------+----------------------
+--------------+-----------------------------------------------------+
```

其中 java 是 Java 外表插件的通用实现，jdbc 是一个 Java JDBC 数据源，基于 java 插件实现，MySQL 是一个MySQL JDBC 数据源，基于 jdbc 插件实现。

3.5 使用示例

步骤一：环境准备

假设有以下数据库环境：

组件	地址	端口	用户	数据库
MySQL 数据库	xxx.xxx.xxx.1	3306	root	test
OceanBase 集群	xxx.xxx.xxx.2	2881	root@mysql001	test

步骤二：创建测试表

1. 在 MySQL 数据库中创建表。

   示例如下：

```
CREATE TABLE `lineitem` (
`l_orderkey` bigint(20) NOT NULL,
`l_partkey` bigint(20) NOT NULL,
`l_suppkey` bigint(20) NOT NULL,
`l_linenumber` bigint(20) NOT NULL,
`l_quantity` bigint(20) NOT NULL,
`l_extendedprice` decimal(10,2) NOT NULL,
`l_discount` decimal(10,2) NOT NULL,
`l_tax` decimal(10,2) NOT NULL,
`l_returnflag` char(1) DEFAULT NULL,
`l_linestatus` char(1) DEFAULT NULL,
`l_shipdate` date DEFAULT NULL,
`l_commitdate` date DEFAULT NULL,
`l_receiptdate` date DEFAULT NULL,
`l_shipinstruct` char(25) DEFAULT NULL,
`l_shipmode` char(10) DEFAULT NULL,
`l_comment` varchar(44) DEFAULT NULL,
PRIMARY KEY (`l_orderkey`,`l_linenumber`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

2. 在 OceanBase 数据库中创建外表。

示例如下：

```
CREATE EXTERNAL TABLE lineitem (
l_orderkey bigint,
l_partkey bigint,
l_suppkey bigint,
l_linenumber bigint,
l_quantity bigint,
l_extendedprice decimal(10,2),
l_discount decimal(10,2),
l_tax decimal(10,2),
l_returnflag char(1) ,
l_linestatus char(1) ,
l_shipdate date ,
l_commitdate date ,
l_receiptdate date ,
l_shipinstruct char(25) ,
l_shipmode char(10) ,
l_comment varchar(44)
)
PROPERTIES (
TYPE='plugin',
NAME='mysql',
PARAMETERS='{"user":"root","password":"","table":"lineitem","jdbc_url":"jdbc:
mysql://xxx.xxx.xxx.1:3306/test?useSSL=false"}'
);
```

注意

在 JDBC 连接串中增加了 `useSSL=false` 参数，因为 MySQL JDBC 插件使用了 8.0.3 版本，连接比较 MySQL 5.7 版本时需要增加此选项。

3. 在 MySQL 数据库表中插入测试数据。

   示例如下：

```
INSERT INTO `lineitem` (`l_orderkey`, `l_partkey`, `l_suppkey`, `l_linenumber`,
`l_quantity`, `l_extendedprice`, `l_discount`, `l_tax`, `l_returnflag`, `l_linestatus`,
`l_shipdate`, `l_commitdate`, `l_receiptdate`, `l_shipinstruct`, `l_shipmode`,
`l_comment`) VALUES
(1, 155190, 7706, 1, 17, 21168.23, 0.04, 0.02, 'N', 'O', '1996-03-13', '1996-02-12',
'1996-03-22', 'DELIVER IN PERSON', 'TRUCK', 'egular courts above the'),
(1, 67310, 7311, 2, 36, 45983.16, 0.09, 0.06, 'N', 'O', '1996-04-12', '1996-02-28',
'1996-04-20', 'TAKE BACK RETURN', 'MAIL', 'ly final dependencies: slyly bold '),
(1, 63700, 3701, 3, 8, 13309.60, 0.10, 0.02, 'N', 'O', '1996-01-29', '1996-03-05',
'1996-01-31', 'TAKE BACK RETURN', 'REG AIR', 'riously. regular, express dep'),
(1, 2132, 4633, 4, 28, 28955.64, 0.09, 0.06, 'N', 'O', '1996-04-21', '1996-03-30',
'1996-05-16', 'NONE', 'AIR', 'lites. fluffily even de'),
(1, 24027, 1534, 5, 24, 22824.48, 0.10, 0.04, 'N', 'O', '1996-03-30', '1996-03-14',
'1996-04-01', 'NONE', 'FOB', ' pending foxes. slyly re'),
(1, 15635, 638, 6, 32, 49620.16, 0.07, 0.02, 'N', 'O', '1996-01-30', '1996-02-07',
'1996-02-03', 'DELIVER IN PERSON', 'MAIL', 'arefully slyly ex'),
(2, 106170, 1191, 1, 38, 44694.46, 0.00, 0.05, 'N', 'O', '1997-01-28', '1997-01-14',
'1997-02-02', 'TAKE BACK RETURN', 'RAIL', 'ven requests. deposits breach a'),
(3, 4297, 1798, 1, 45, 54058.05, 0.06, 0.00, 'R', 'F', '1994-02-02', '1994-01-04',
'1994-02-23', 'NONE', 'AIR', 'ongside of the furiously brave acco'),
(3, 19036, 6540, 2, 49, 46796.47, 0.10, 0.00, 'R', 'F', '1993-11-09', '1993-12-20',
'1993-11-24', 'TAKE BACK RETURN', 'RAIL', ' unusual accounts. eve'),
(3, 128449, 3474, 3, 27, 39890.88, 0.06, 0.07, 'A', 'F', '1994-01-16', '1993-11-22',
'1994-01-23', 'DELIVER IN PERSON', 'SHIP', 'nal foxes wake. ');
```

4. 在 OceanBase 数据库中查询数据。

   示例如下：

```
SELECT * FROM lineitem;
```

返回结果如下：

```
+------------+-----------+-----------+-------------+-----------+----------------
+------------+-------+-------------+-------------+-----------+-------------
+--------------+-----------------+------------+-----------------------------------
+
| l_orderkey | l_partkey | l_suppkey | l_linenumber | l_quantity | l_extendedprice |
l_discount | l_tax | l_returnflag | l_linestatus | l_shipdate | l_commitdate |
l_receiptdate | l_shipinstruct | l_shipmode | l_comment |
+------------+-----------+-----------+-------------+-----------+----------------
+------------+-------+-------------+-------------+-----------+-------------
+--------------+-----------------+------------+-----------------------------------
+
| 1 | 155190 | 7706 | 1 | 17 | 21168.23 | 0.04 | 0.02 | N | O | 1996-03-13 | 1996-02-12 |
1996-03-22 | DELIVER IN PERSON | TRUCK | egular courts above the |
| 1 | 67310 | 7311 | 2 | 36 | 45983.16 | 0.09 | 0.06 | N | O | 1996-04-12 | 1996-02-28 |
1996-04-20 | TAKE BACK RETURN | MAIL | ly final dependencies: slyly bold |
| 1 | 63700 | 3701 | 3 | 8 | 13309.60 | 0.10 | 0.02 | N | O | 1996-01-29 | 1996-03-05 |
1996-01-31 | TAKE BACK RETURN | REG AIR | riously. regular, express dep |
| 1 | 2132 | 4633 | 4 | 28 | 28955.64 | 0.09 | 0.06 | N | O | 1996-04-21 | 1996-03-30 |
1996-05-16 | NONE | AIR | lites. fluffily even de |
| 1 | 24027 | 1534 | 5 | 24 | 22824.48 | 0.10 | 0.04 | N | O | 1996-03-30 | 1996-03-14 |
1996-04-01 | NONE | FOB | pending foxes. slyly re |
| 1 | 15635 | 638 | 6 | 32 | 49620.16 | 0.07 | 0.02 | N | O | 1996-01-30 | 1996-02-07 |
1996-02-03 | DELIVER IN PERSON | MAIL | arefully slyly ex |
| 2 | 106170 | 1191 | 1 | 38 | 44694.46 | 0.00 | 0.05 | N | O | 1997-01-28 | 1997-01-14 |
1997-02-02 | TAKE BACK RETURN | RAIL | ven requests. deposits breach a |
| 3 | 4297 | 1798 | 1 | 45 | 54058.05 | 0.06 | 0.00 | R | F | 1994-02-02 | 1994-01-04 | 1994-
02-23 | NONE | AIR | ongside of the furiously brave acco |
| 3 | 19036 | 6540 | 2 | 49 | 46796.47 | 0.10 | 0.00 | R | F | 1993-11-09 | 1993-12-20 |
```

```
1993-11-24 | TAKE BACK RETURN | RAIL | unusual accounts. eve |
| 3 | 128449 | 3474 | 3 | 27 | 39890.88 | 0.06 | 0.07 | A | F | 1994-01-16 | 1993-11-22 |
1994-01-23 | DELIVER IN PERSON | SHIP | nal foxes wake. |
+-----------+----------+----------+-------------+-----------+----------------
+-----------+-------+-------------+-------------+-----------+-------------
+--------------+------------------+-----------+-----------------------------------
+
10 rows in set
```

说明

由于需要加载 jar 包等资源，首次运行时比较慢。

3.6 数据类型

MySQL 数据库中有丰富的数据类型，不过 OceanBase 数据库的 MySQL 外表插件还未完全支持。当前版本 OceanBase 数据库的 MySQL 外表插件支持的数据类型如下所示：

- 整数类型：TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT（含 UNSIGNED）。
- 浮点数：FLOAT、DOUBLE、DECIMAL。
- 字符串类型：CHAR、VARCHAR。
- 文本类型：TEXT、CLOB。
- 二进制类型：BINARY、VARBINARY、BLOB。
- 日期时间：DATE、TIME、DATETIME、TIMESTAMP、YEAR。
- 布尔类型：BOOLEAN。
- 空间类型：GIS 相关。
- 其它类型：JSON、ENUM、SET。

3.7 谓词下推规则

为了提高数据的访问效率，OceanBase 数据库支持部分谓词下推到远程 MySQL 数据库。

说明

这里的谓词就是 SQL 中的 `WHERE` 条件，有些地方写作 predicate 或 filter。

支持下推的谓词的条件：

- 数据类型：整数、浮点数（部分场景）、字符串（不包括 `CHAR` 类型）。
- 运算符：`=`、`!=(<>)`、`<=`、`<`、`>=`、`>`、`IS [NOT] NULL`、`[NOT] IN`、`[NOT] LIKE`、`[NOT] BETWEEN`。
- 组合表达式：`AND`、`OR`、`NOT`。

3.8 系统升级

对于已有的 OceanBase 集群，可以采取逐个节点重启的办法安装或升级插件。

建议升级步骤如下：

1. 将新版本的插件 jar 包放到指定目录（配置项 `ob_java_connector_path` 指向的目录）。
2. 逐个重启节点，重启时检查节点加载插件情况。
3. OceanBase 集群重启完成，检查集群插件安装情况。

关于集群升级的信息，参见 升级 OceanBase 集群。

3.9 相关文档

关于创建外表语法的详细介绍，参见 CREATE EXTERNAL TABLE。