Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequent
ial, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please contact Alibaba Cloud directly if you discover any errors in this document.
# Document conventions

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔴</td>
<td>A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.</td>
<td>Danger: Resetting will result in the loss of user configuration data.</td>
</tr>
<tr>
<td>⚠️</td>
<td>A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.</td>
<td>Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.</td>
</tr>
<tr>
<td>!</td>
<td>A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.</td>
<td>Notice: If the weight is set to 0, the server no longer receives new requests.</td>
</tr>
<tr>
<td>📝</td>
<td>A note indicates supplemental instructions, best practices, tips, and other content.</td>
<td>Note: You can use Ctrl + A to select all files.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Closing angle brackets are used to indicate a multi-level menu cascade.</td>
<td>Click Settings &gt; Network &gt; Set network type.</td>
</tr>
<tr>
<td>**</td>
<td>Bold formatting is used for buttons, menus, page names, and other UI elements.</td>
<td>Click OK.</td>
</tr>
<tr>
<td>`</td>
<td>Courier font is used for commands.</td>
<td>Run the <code>cd /d C:/windows</code> command to enter the Windows system folder.</td>
</tr>
<tr>
<td>*</td>
<td>Italic formatting is used for parameters and variables.</td>
<td><code>bae log list --instanceid Instance_ID</code></td>
</tr>
<tr>
<td>[]</td>
<td>This format is used for an optional value, where only one item can be selected.</td>
<td>`ipconfig [-all</td>
</tr>
<tr>
<td>Style</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>{} or {a</td>
<td>b}</td>
<td>This format is used for a required value, where only one item can be selected.</td>
</tr>
</tbody>
</table>
Contents

Legal disclaimer

Document conventions

1 Workbench

1.1 Data integration overview

1.2 Terms

1.3 Create a Data Integration task

1.4 Supported data sources

1.5 Test data source connectivity

1.6 Task configuration

1.6.1 Data synchronization task configuration

1.6.2 Wizard mode configuration

1.6.3 Script mode configuration

1.6.4 Configure reader plug-in

1.6.4.1 Configure DRDS Reader

1.6.4.2 Configure HBase reader

1.6.4.3 Configuring HDFS Reader

1.6.4.4 Configure MaxCompute Reader

1.6.4.5 Configure MongoDB Reader

1.6.4.6 Configure DB2 reader

1.6.4.7 Configure MySQL Reader

1.6.4.8 Configure Oracle Reader

1.6.4.9 Configure OSS Reader

1.6.4.10 Configuring FTP Reader

1.6.4.11 Configure Table Store (OTS) Reader

1.6.4.12 Configuring PostgreSQL Reader

1.6.4.13 Configuring SQL server Reader

1.6.4.14 Configure LogHub Reader

1.6.4.15 Configure OTSReader-Internal

1.6.4.16 Configure OTSStream Reader

1.6.4.17 Configure RDBMS Reader

1.6.4.18 Configure Stream Reader

1.6.4.19 Configure HybridDB for MySQL Reader

1.6.4.20 Configure AnalyticDB for PostgreSQL Reader

1.6.4.21 Configure POLARDB Reader

1.6.5 Configure writer plug-in

1.6.5.1 Configure AnalyticDB for MySQL 2.0 Writer

1.6.5.2 Configure DataHub Writer

1.6.5.3 Configure DB2 Writer

1.6.5.4 Configure DRDS Writer

1.6.5.5 Configure FTP Writer
1.6.5.6 Configure HBase Writer......................................................... 213
1.6.5.7 Configure HBase11xsql Writer................................................219
1.6.5.8 Configure HDFS Writer......................................................... 222
1.6.5.9 Configure MaxCompute Writer.............................................. 231
1.6.5.10 Configure Memcache (OCS) Writer....................................... 238
1.6.5.11 Configure MongoDB Writer................................................ 242
1.6.5.12 Configure MySQL Writer..................................................... 245
1.6.5.13 Configuring Oracle Writer.................................................. 252
1.6.5.14 Configure OSS Writer....................................................... 257
1.6.5.15 Configure PostgreSQL Writer............................................. 264
1.6.5.16 Configure Redis Writer..................................................... 269
1.6.5.17 Configure SQL Server Writer............................................. 278
1.6.5.18 Configure Elasticsearch Writer......................................... 283
1.6.5.19 Configure LogHub Writer................................................ 288
1.6.5.20 Configure OpenSearch Writer.......................................... 290
1.6.5.21 Configure Table Store (OTS) Writer.................................... 295
1.6.5.22 Configure RDBMS Writer.................................................... 299
1.6.5.23 Configure Stream Writer................................................. 304
1.6.5.24 Configure HybridDB for MySQL Writer.............................. 306
1.6.5.25 Configure HybridDB for PostgreSQL Writer......................... 312
1.6.5.26 Configure POLARDB Writer............................................. 317
1.6.6 Optimizing configuration.................................................... 324

2 Data source configuration.......................................................... 331
  2.1 Data source isolation................................................................. 331
  2.2 Configure AnalyticDB for MySQL 2.0 data source........................335
  2.3 Configure SQL Server data source............................................ 337
  2.4 Configure MongoDB data source............................................ 343
  2.5 DataHub data source.............................................................. 349
  2.6 Configure the DM data source.................................................. 352
  2.7 Configure DRDS data sources................................................ 356
  2.8 Configure FTP data source..................................................... 359
  2.9 Configuring HDFS data source................................................ 364
  2.10 Add LogHub data source..................................................... 367
  2.11 Configure MaxCompute data source....................................... 370
  2.12 Configure Memcache data source......................................... 373
  2.13 Configure MySQL data source............................................. 376
  2.14 Configure Oracle data source.............................................. 382
  2.15 Configure OSS data source................................................ 386
  2.16 Configure Table Store (OTS) data source............................... 389
  2.17 Configure PostgreSQL data source........................................ 392
  2.18 Configure Redis data source............................................... 398
  2.19 Configure HybridDB for MySQL data source........................... 402
  2.20 Configure HybridDB for PostgreSQL data source...................... 405
  2.21 Configure a POLARDB data source....................................... 409
1 Workbench

1.1 Data integration overview

The Alibaba Cloud Data Integration is a data synchronization platform that provides stable, efficient, and elastically scalable services. Data integration is designed to implement fast and stable data migration and synchronization between multiple heterogeneous data sources in complex network environments.

Offline (batch) data synchronization

The offline (batch) data channel provides a set of abstract data extraction plug-ins (Readers) and data writing plug-ins (Writers) by defining the source and target databases and datasets. Also, it designs a set of simplified intermediate data transmission formats based on the framework to transfer data between any structured and semi-structured data sources.

Supported data source types

Data integration supports diverse data sources as follows:

- Text storage (FTP, SFTP, OSS, Multimedia files),
- Database (RDS, DRDS, MySQL, PostgreSQL),
- NoSQL (Memcache, Redis, MongoDB, HBase),
- Big data (MaxCompute, AnalyticDB, HDFS),
- MPP database (HybridDB for MySQL).

For more information, see Supported data sources.

Note:
The data sources configured information varies greatly from each other, and the parameter configuration information must be queried in detail based on the actual scenario. For this reason, the detailed parameter descriptions are available on the data source configuration and job configuration pages, which can be queried and used as needed.

Synchronous development description

Synchronous development provides both wizard and script modes.
Wizard: Provides a visualized development guide and comprehensive details about data sync task configuration. This mode is cost-effective, but lacks certain advanced functions.

Script: Allows you to directly write a data sync JSON script for completing the data sync development. It is suitable for advanced users, but has a high learning cost. It also provides diverse and flexible functions for delicacy configuration management.

Note:

- The code generated in wizard mode can be converted to script mode code. The code conversion is unidirectional, and cannot be converted back to wizard mode format. This is because the script mode capabilities are a superset of the wizard mode.
- Always configure the data source and create the target table before writing codes.

Description of network types

The networks can be classified as classic network, VPC network, and local IDC network (planning).

- Classic network: A network that is centrally deployed on the Alibaba Cloud public infrastructure network planned and managed by Alibaba Cloud. This network type suits customers that have ease-of-use requirements.
- VPC network: An isolated network environment created on Alibaba Cloud. In this network type, you have full control over the virtual network, including customizing the IP address range, partitioning network segments, and configuring routing tables and gateways.
- Local IDC network: The network environment of your server room, which is isolated from the Alibaba Cloud network.

See classic network and VPC FAQ page for questions related to classic and VPC networks.

Note:

- The public network access is supported. The public network access only selects the classic network as the network type. Note the public network bandwidth speed and relevant network traffic charges when using this network type. We do not recommend this configuration except in special cases.
• Network connections are planned for data synchronization, you can use the locally added resource + Script Mode scheme for synchronous data transfer, you can also use the Shell + DataX scheme.

• The Virtual Private Cloud (VPC) creates an isolated network environment that allows you to customize the IP address range, network segments, and gateways. The VPC applications have expanded the scope of VPC security, as a result data integration provides RDS for MySQL, RDS for SQL Server, and RDS for PostgreSQL and eliminates the need to purchase extra ECSs that reside on the same network as the VPC. Instead, the system guarantees interconnectivity by detecting devices automatically through the reverse proxy. The VPC supports other Alibaba Cloud databases including PPAS, OceanBase, Redis, MongoDB, Memcache, TableStore, and HBase. For any non-RDS data sources, an ECS on the same network is required for configuring data integration synchronization tasks on the VPC network and ensuring interconnectivity.

Limits

• Supports the following data synchronization types: structured (such as RDS and DRDS), semi-structured, and non-structured, such as OSS and TXT. The specified synchronization data must be abstracted as structured data. That is, data integration supports data synchronization that can transmit data that can be abstracted to a logical two-dimensional table, other fully unstructured data, such as a MP3 section stored in OSS. Data integration does not support synchronizing dataset to MaxCompute, which is still in development.

• Supports data synchronization and exchange between single region and cross-region data storage.

For certain regions, cross-region data transmission is supported, but not guaranteed by the classic network. If you need to use this function, while the tested classic network is disconnected, consider using the public network connection instead.

• Only data synchronization (transmission) is performed and no consumption plans of data stream is provided.

References

• For a detailed description of data synchronization task configuration, see create a data synchronization task.
• For a detailed introduction to processing unstructured data such as OSS, see access OSS unstructured data.

1.2 Terms

DMU

Data Migration Unit (DMU) is used to measure the amount of resources consumed by data integration, including CPU, memory, and network. One DMU represents the minimum amount of resources used for a data synchronization task.

Slot

By default, the resource group provides you 50 slots and each DMU occupies 2 slots. This means the default resource group supports 25 DMUs at the same time. You can submit a ticket to apply for more slots in the default resource group.

Number of concurrencies

Concurrency indicates the maximum number of threads used to concurrently read or write data in the data storage of a data synchronization task.

Speed limit

The speed limit indicates the maximum speed of synchronization tasks.

Dirty data

Dirty data indicates invalid or incorrectly formatted data. For example, if the source has varchar type data, but is written to a destination column as an int type data. If a data conversion exception occurs, the data cannot be written to the destination column.

Data sources

The data source processed by DataWorks can be from a database or a data warehouse. DataWorks supports various data source types, and supports different data source conversions.

1.3 Create a Data Integration task

This topic describes how to create a Data Integration task.

• Data Integration is a reliable, secure, cost-effective, and elastically scalable data synchronization platform provided by Alibaba Group. It can be used across
heterogeneous data storage systems and provides full or incremental data access channels in different network environments for a variety of data sources.

- A reader plug-in reads data from a database at the underlying layer by connecting to a remote database and running SQL statements to select data from the database.
- A writer plug-in writes data into a database at the underlying layer by connecting to a remote database and running SQL statements to write data into the database.

Preparations

Create an Alibaba Cloud account

1. Activate an Alibaba Cloud account, and create the AccessKeys for this account.
2. Activate MaxCompute to automatically generate a default MaxCompute data source, and log on to DataWorks using the Alibaba Cloud account.
3. Create a workspace. You can collaboratively complete workflows and maintain data or tasks in the workspace. Before using DataWorks, you need to create a workspace.

Note:

You can grant RAM accounts the permissions to create Data Integration tasks. For more information, see Create a RAM account.

Create source and destination databases and tables

1. You can create tables by running statements or create tables directly in the data source client. For more information about how to create databases and tables of different data source types, see their official documents.
2. Grant read and write permissions on the databases and tables.

Note:

Generally, a reader plug-in requires at least the read permission, while a writer plug-in requires the add, delete, and modify permissions. We recommend that you grant sufficient permissions on tables of databases in advance.

Procedure

Create a data source

1. Obtain data source information about a database.
2. Configure the data source on the GUI.

Note:

- Not all data sources can be configured on the GUI. If you cannot find the configuration page for a data source, you can configure it in script mode by writing data source information in a JSON script.
- For more information about the data sources that are supported, see Supported data sources.

(Optional) Create a custom resource group

1. Create a resource group.
2. Add a server.
3. Install the agent.
4. Test the connectivity.

Note:

- If the data source is located in a private network environment or the resources provided by DataWorks do not meet your requirements, you can create a custom resource group.
- We recommend that you set the network type of the custom resource group to VPC regardless whether the server is in a classic network or VPC.
- For more information about how to configure a custom resource group, see Add scheduling resources.
- Best practices:
  - Data synchronization when either the source or destination is located in a private network environment
  - Data synchronization when both the source and destination are located in a private network environment

Configure a Data Integration task

1. Configure the reader of the Data Integration task. For more information about how to configure a reader, see Configure a reader plug-in.
2. Configure the writer of the Data Integration task. For more information about how to configure a writer, see Configure a writer plug-in.
3. Configure the mapping between the reader and writer.

4. Configure channel control. You can switch to a custom resource group in this step.

Note:
- A task can be configured in wizard or script mode.
- When configuring a task, you can optimize the task speed. For more information, see Optimizing configuration.
- You can switch from the wizard mode to script mode, but not from the script mode to wizard mode. We have provided templates for all plug-ins.

Run the Data Integration task

1. You can run the Data Integration task directly on the GUI. Logs will not be saved.
2. Before submitting the task, you need to configure scheduling. Generally, an instance is generated on the next day after submission.

Note:
When configuring the task, you can set scheduling parameters.

View run logs

You can view run logs of your task in O&M.

Note:
You can find the DAG in O&M, right-click the DAG, and select Run Log to view the run logs.

1.4 Supported data sources

Data Integration is a stable, efficient, and elastically scalable data synchronization platform that Alibaba Group provides to external users. It provides offline (batch) data access channels for Alibaba Cloud's big data computing engines, including MaxCompute, AnalyticDB for MySQL 2.0, and Object Storage Service (OSS).

The following table lists data source types supported by data integration:
<table>
<thead>
<tr>
<th>Data source category</th>
<th>Data source type</th>
<th>Extraction (reader)</th>
<th>Import (writer)</th>
<th>Supported methods</th>
<th>Supported types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational databases</td>
<td>MySQL</td>
<td>Yes. For more information, see Configure MySQL reader</td>
<td>Yes. For more information, see Configure MySQL writer</td>
<td>Wizard and script</td>
<td>Alibaba Cloud and on-premise</td>
</tr>
<tr>
<td>Relational databases</td>
<td>SQL Server</td>
<td>Yes. For more information, see Configure SQL server reader</td>
<td>Yes. For more information, see Configure SQL server writer</td>
<td>Wizard and script</td>
<td>Alibaba Cloud and on-premise</td>
</tr>
<tr>
<td>Relational database</td>
<td>PostgreSQL</td>
<td>Yes. For more information, see Configure PostgreSQL reader</td>
<td>Yes. For more information, see Configure PostgreSQL writer</td>
<td>Wizard and script</td>
<td>Alibaba Cloud and on-premise</td>
</tr>
<tr>
<td>Relational databases</td>
<td>Oracle</td>
<td>Yes. For more information, see Configure Oracle reader</td>
<td>Yes. For more information, see Configure Oracle writer</td>
<td>Wizard and script</td>
<td>On-premise</td>
</tr>
<tr>
<td>Relational databases</td>
<td>DRDS</td>
<td>Yes. For more information, see Configure DRDS reader</td>
<td>Yes. For more information, see Configure DRDS writer</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Relational databases</td>
<td>DB2</td>
<td>Yes. For more information, see Configure DB2 reader</td>
<td>Yes. For more information, see Configure DB2 writer</td>
<td>Script</td>
<td>On-premise</td>
</tr>
<tr>
<td>Relational databases</td>
<td>DM</td>
<td>Yes</td>
<td>Yes</td>
<td>Script</td>
<td>On-premise</td>
</tr>
<tr>
<td>Relational databases</td>
<td>RDS for PPAS</td>
<td>Yes</td>
<td>Yes</td>
<td>Script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Data source category</td>
<td>Data source type</td>
<td>Extraction (reader)</td>
<td>Import (writer)</td>
<td>Supported methods</td>
<td>Supported types</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>MPP</td>
<td>HybridDB for MySQL</td>
<td>Yes</td>
<td>Yes</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>MPP</td>
<td>HybridDB for PostgreSQL released</td>
<td>Yes</td>
<td>Yes</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Big data storage</td>
<td>MaxCompute</td>
<td>Yes. For more information, see Configure MaxCompute reader.</td>
<td>Yes. For more information, see Configure MaxCompute writer.</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Big data storage</td>
<td>DataHub</td>
<td>No</td>
<td>Yes. For more information, see Configure DataHub writer.</td>
<td>Script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Big data storage</td>
<td>ElasticSearch</td>
<td>No</td>
<td>Yes. For more information, see Configure ElasticSearch writer.</td>
<td>Script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Big data storage</td>
<td>AnalyticDBAnalyticDB for MySQL 2.0</td>
<td>Yes</td>
<td>Yes</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Unstructured storage</td>
<td>OSS</td>
<td>Yes. For more information, see Configure OSS reader.</td>
<td>Yes. For more information, see Configure OSS writer.</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>Unstructured storage</td>
<td>HDFS</td>
<td>Yes For more information, see Configure HDFS reader.</td>
<td>Yes For more information, see Configure HDFS writer.</td>
<td>Script</td>
<td>On-premise</td>
</tr>
<tr>
<td>Data source category</td>
<td>Data source type</td>
<td>Extraction (reader)</td>
<td>Import (writer)</td>
<td>Supported methods</td>
<td>Supported types</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Unstructured storage</td>
<td>FTP</td>
<td>Yes. For more information, see Configure FTP reader.</td>
<td>Yes. For more information, see Configure FTP writer.</td>
<td>Wizard and script</td>
<td>On-premise</td>
</tr>
<tr>
<td>Message queue</td>
<td>LogHub</td>
<td>Yes. For more information, see Configure LogHub reader.</td>
<td>Yes. For more information, see Configure LogHub writer.</td>
<td>Wizard and script</td>
<td>Alibaba Cloud</td>
</tr>
<tr>
<td>NoSQL</td>
<td>HBase</td>
<td>Yes. For more information, see Configure HBase reader.</td>
<td>Yes. For more information, see Configure HBase writer.</td>
<td>Script</td>
<td>Alibaba Cloud and on-premise</td>
</tr>
<tr>
<td>NoSQL</td>
<td>MongoDB</td>
<td>Yes. For more information, see Configure MongoDB reader.</td>
<td>Yes. For more information, see Configure MongoDB writer.</td>
<td>Script</td>
<td>Alibaba Cloud and on-premise</td>
</tr>
<tr>
<td>NoSQL</td>
<td>Memcache</td>
<td>No</td>
<td>Yes. For more information, see Configure Memcache (OCS) writer.</td>
<td>Script</td>
<td>Alibaba Cloud and on-premise Memcache</td>
</tr>
<tr>
<td>NoSQL</td>
<td>Table Store (corresponding data source name: OTS)</td>
<td>Yes. For more information, see Configure Table Store(OTS) reader.</td>
<td>Yes. For more information, see Configure Table Store (OTS) writer.</td>
<td>Script</td>
<td>Alibaba Cloud</td>
</tr>
</tbody>
</table>
### 1.5 Test data source connectivity

<table>
<thead>
<tr>
<th>Data source</th>
<th>Data source type</th>
<th>Network type</th>
<th>Supports test connectivity?</th>
<th>Add custom resource group</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With public IP address</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Without public IP address</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>On-premise ECS</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Classic network</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL Server</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With public IP address</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Without public IP address</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data source (for MaxCompute data sources)</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Data source type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>With public IP address</td>
<td>Yes</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without public IP address</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On-premise ECS</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>With public IP address</td>
<td>Yes</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without public IP address</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On-premise ECS</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DRDS</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HybridDB for MySQL</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HybridDB for PostgreSQL, released</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Data source type</td>
<td>Data source type</td>
<td>Network type</td>
<td>Supports test connectivity?</td>
<td>Add custom resource group</td>
</tr>
<tr>
<td>DataWorks</td>
<td>Data Aggregation / 1 Workbench</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data source</td>
<td>Data source type</td>
<td>Network type</td>
<td>Supports test connectivity?</td>
<td>Add custom resource group</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>-----------------------</td>
<td>-----------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>AnalyticDB for MySQL 2.0</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
</tr>
<tr>
<td>OSS</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>HDFS</td>
<td>With public IP address</td>
<td>Yes</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On-premise ECS</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>FTP</td>
<td>With public IP address</td>
<td>Yes</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Without public IP address</td>
<td>No</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On-premise ECS</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>MongoDB</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>With public IP address</td>
<td>Yes</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On-premise ECS</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Memcache</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
</tr>
<tr>
<td>Redis</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>With public IP address</td>
<td>Yes</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
## Data source and connectivity test

<table>
<thead>
<tr>
<th>Data source type</th>
<th>Network type</th>
<th>Supports test connectivity?</th>
<th>Add custom resource group</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-premise ECS</td>
<td>Classic network</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Table Store (for OTS data sources)</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>Under development</td>
<td>Yes</td>
</tr>
<tr>
<td>DataHub</td>
<td>ApsaraDB</td>
<td>Classic network</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>VPC network</td>
<td>No</td>
<td>-</td>
</tr>
</tbody>
</table>

### Description

In the preceding table, "-" means this item is unavailable. "No" means the connectivity test failed and a custom resource group must be added, and the synchronization task can be configured.

#### Data sources in VPC environment:

- Connectivity tests for RDS data sources in VPC environment is supported.
- Other data sources in VPC environment are under development.
- Financial Cloud networks does not support connectivity tests.

#### User-created ECS data sources:

- The classic network typically supports JDBC-based connectivity tests on the public network.
- The VPC does not support connectivity tests for now.
- Currently, cross-region sources connectivity tests is not supported.
- Financial Cloud networks do not support connectivity tests.

Currently, data synchronization is implemented solely by adding a custom resource group.
For created ECS data sources, add the scheduling cluster IP address to the security group for both inbound and outbound traffic in the public network and classic network. If the security group is not added, a disconnection error may occur during synchronization. For more information, see #unique_73.

You cannot add an extensive port range on the ECS security group page. To add them, use the security group API of ECS. For more information, see `AuthorizeSecurityGroup`.

- Data sources created in local IDCs or on the ECS server without public IP addresses:
  - Connectivity tests are not supported.
  - A custom resource group must be added for configuring the synchronization tasks.

- Public-network-based JDBC is applied to data sources created in local IDCs or on the ECS server with public IP addresses for connectivity tests. If the connectivity test fails, check the limits of the local network or relevant databases.

**Note:**

The following example describes the billing of synchronizing data from RDS to MaxCompute:

Currently, data integration is free of charge, but you might still be billed for certain products. Configuring MaxCompute data synchronization in DataWorks is free of charge, but you will be billed for manually adding the parameter in the script mode to set a public IP address for the MaxCompute tunnel. However, this parameter is unavailable in the template generated in the script mode.

**Conclusion**

When a connectivity test fails, you need to verify the data source region, network type, and whether the full instance ID, database name and user name are valid in the RDS whitelist. Examples of common errors are as follows:
The Database Password is invalid as follows:

![Database Password Error]

The network connection failed as follows:

```
"com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure"
```

The network disconnected during synchronization or because of other conditions.

View the full log to locate the scheduled resource and to determine whether it is a custom resource.

If so, check whether the IP address of the custom resource group has been added to the data source whitelist, such as RDS. This also applies to MongoDB.

Check whether connectivity tests between both data sources was successful and if their whitelists are complete. The test result will vary, if the whitelists are incomplete. Specifically, the test is successful if the task is assigned to the added scheduling server or failed if no scheduling server has been added.

For tasks that are successful, but the disconnection error 8000 is found in the log:

This condition occurs when the custom scheduling resource group is used and the IP address 10.116.134.123 and port 8000 does not have security group
inbound traffic permission. Under this condition, add the IP address and the port, and run the task again.

Connectivity test exception examples

• Example 1

A database test connection error occurred resulting in a data source connectivity test exception. The database connection string is "jdbc:mysql://xx.xx.xx.xx:xxxx/t_uoer_brade", the user name is "xxxx_test", and the exception message is "Access denied for user "xxxx_test"@"%" to database "yyyy_demo".

• Troubleshooting

1. Check if the entered information is valid.
2. Check if the password, whitelist, or your account has permission to access the database. You can add the required permissions in the RDS console.

• Example 2

A test connection exception occurred resulting in the data source connectivity test exception. The displayed error message is as follows:

```
error message: Timed out after 5000 ms while waiting for a server that matches ReadPreferenceServerSelector{readPreference=primary}. Client view of cluster state is {type=UNKNOWN, servers=[{(xxxxxxxxxxxx), type=UNKNOWN, state=CONNECTING, exception={com.mongodb.MongoSocketReadException: Prematurely reached end of stream}}]}
```

- Troubleshooting

If you are using MongoDB without VPC connection. You must add a whitelist for the connectivity test of the MongoDB data source. For more information, see #unique_74.

1.6 Task configuration

1.6.1 Data synchronization task configuration

1.6.2 Wizard mode configuration

This topic describes how to configure tasks through the Data Integration Wizard mode.

The steps for task configuration are as follows:

1. Create a data source.
2. Create a synchronization task and configure the synchronization task reader.
3. Configure the synchronization task writer.
4. Configure the mapping between the synchronization task reader and the synchronization task writer.
5. Configure the concurrency, transmission rate, dirty data records, resource groups, and other information of the synchronization task.
6. Configure the scheduling attribute of the synchronization task.

Note:
The following is an introduction to specific operation step implementation, where each of the following steps directs to the corresponding topic. When you complete a step, to continue onto the next step click the link to return to this topic.

Create a synchronization task and reader

Note:

- Data Integration does not support test connectivity for certain data sources. For more information on data source test connectivity, see Test data source connectivity.
- Data sources are frequently created locally and cannot be connected without a public network IP address or network. In this case, testing connectivity at the time of the data source configuration might fail. Data Integration supports #unique_12 to solve network inaccessibility.

Create data source

Synchronization tasks support data transmission between different homogenous and heterogeneous data sources. First, register the target data source in Data Integration. Then you can select the data source directly when configuring a synchronization task on Data Integration. For more information on the synchronous data source types supported by Data Integration, see Supported data sources.

Confirming the target data source is supported by Data Integration, after you can register the data source in Data Integration. For more information on data source registration, see Configuring data source information.
This topic mainly describes how to synchronize task configuration in Wizard Mode. Select Wizard Mode when creating new synchronization tasks.

1. Enter the DataWorks management console as a developer, and click Data Development in the corresponding project Action column.
2. Click Data Development in the left-hand navigation pane to open the Business Process navigator.
3. Right-click Business Flow in the navigation pane to create Data Integration Node > Data Sync, and enter the synchronization task's name.

4. After creating the synchronization task, you can manually configure the reader data source and the target table information for the data synchronization task.
For more information on how to select a data source to read from, see Configuring Reader.

Note:
Incremental data synchronization is required for many tasks when configuring read-side data sources. You can now obtain relative date in conjunction with #unique_18 to complete the requirement to obtain the incremental data.

Configure the Writer

After the reader data source is configured, you can manually configure the Writer data source and the target table information for the data synchronization task. When you are selecting the data source to write on, see Configure Writer.

Note:
You need to select a Write Mode based on data sources, such as overwrite mode or append mode for most tasks. For students with write control requirements, refer to the Configure Writer documentation to select the Write Mode.
Configure mapping

When you complete the configuration for both read/write, you need to specify a mapping relationship between the read/write end columns, and select Map of the same name or Enable same-line mapping.

- Enable same-line mapping: Automatically sets the mapping relationship for the same row of data.
- Automatic layout: The field order is displayed after the mapping relationship is set.

![Mapping Image]

Note:
The field types mapped between columns should be data compatible.

Task synchronization channel configuration

When the preceding steps are configured, the efficiency configuration is required. The efficiency configuration mainly includes DMU settings, synchronous concurrency number settings, synchronous rate settings, synchronous dirty data settings and synchronize information, such as Resource Group settings.

![Channel Image]

Parameters:

- DMU: The billing unit for data integration.

Note:
When you set up a DMU, please note the DMU value limits the maximum number of concurrency. Please configure accordingly.

- When you configure the Synchronization Concurrency, the data records are separated into several tasks based on the specified reader shard key. These tasks run simultaneously to improve the transmission rate.

- Synchronous rate: Setting the synchronous rate protects the read-side database from fast extraction speed, and places too much pressure on the source library. It is recommended to throttle the synchronization rate and configure the extraction rate based on the source database configurations.

- For example, if the source has varchar type data, but is written to a destination column with INT type data. A data conversion exception occurs, and the data cannot be written to the destination column. The dirty data is mainly set to control the synchronized data quality. You should set the number of dirty data based on business requirements.

- When you configure a synchronization task, you specify the Resource Group in which the task runs. By default, the synchronization task runs on the Default Resource Group. When the project has a tight schedule of resources, you can also expand a scheduled resource by adding a Custom Resource Group. The synchronization task is then specified to run on a Custom Resource Group. For more information, see Add scheduling resources. You can make configurations based on data source network conditions, project scheduling resource conditions, and business importance.

**Note:**

When synchronizing data is inefficient, see Optimizing configuration to optimize your synchronization tasks.

Scheduling parameters

Use scheduling parameters to filter synchronization task data. The following figure shows how to configure scheduling parameters in the synchronization task.
In the preceding figure, you can declare a schedule parameter variable in the form of a `${variable name}`. When the variable declaration is complete, write the initialization value of the variable in the scheduled parameter properties, the value initialized here by the variable is represented with a dollar sign ($). The content can either be a time expression or a constant.

For example, `${today}` was written in code, by assigning today = $ [yyyymmdd] in the scheduling parameter, you can obtain the current date. For more information on how to add or minus a date, see #unique_18.

Using custom schedule parameters in synchronization tasks

Declare the following parameters in the code of your synchronization task.

- **bizdate**: Obtain business date, and run date-1.
- **cyctime**: Obtain the current run time, in the form of yyyymmddhhmmiss.
- DataWorks provides two system default scheduling parameters: bizdate and cyctime.
Configure scheduling properties

You can set the synchronization task run cycle, run time, task dependency, and more in the scheduling properties. Because the synchronization task starts the ETL task, there are no upstream nodes. It is recommended to use the project root node for upstream settings.

Save the node after completing the synchronization task configuration, and click Submit.

1.6.3 Script mode configuration

This topic describes how to configure tasks through the data integration Script mode.

The task configuration steps are as follows:

1. Create a data source.
2. Create a synchronization task.
3. Import a template.
4. Configure the synchronization task reader.
5. Configure the synchronization task writer.
6. Configure the mapping between the synchronization task reader and the synchronization task writer.
7. Configure the DMUs, concurrency, transmission rates, dirty data records, resource groups, and other synchronization task information.
8. Configure the scheduling attribute of the synchronization task.

Note:
The following introduces the specific implementation of operation steps, each of the following steps jumps to the corresponding topic. After completing the current step, click the link to return to this article to go on to the next step.
Create data source

Synchronization tasks support data transmission between various homogenous and heterogeneous data sources. You need to register the target data source in Data Integration, and then you can select the data source when configuring a synchronization task on Data Integration. Integrate data source types that support synchronization as shown in Supported data sources.

After confirming the target data source is supported by Data Integration, you can register the data source in Data Integration. For detailed data source registration, see Configuring data source information.

Note:

- For some data sources, Data Integration does not support test connectivity. For more information on data source test connectivity, see Test data source connectivity.
- Data sources created locally frequently cannot without a network connection or public network IP address. In this case, testing connectivity during the configuration time of the data source fails directly. Data Integration supports #unique_12 to solve this type of network inaccessibility.

Create a synchronization task and the synchronization task reader

Note:

This topic describes the configuration of synchronization tasks in script mode, select Script Mode when creating new synchronization tasks in dataset generation.

1. Enter the DataWorks management console as a developer, and click Data Development in the corresponding project Action bar.
2. Click Data Development in the left-side navigation pane to open the Business Process.

3. Right-click Business Flow in the left-side navigation pane to create Data Integration > Data Sync, and enter the synchronization Task Name.
4. After creating the synchronization node, click the Switch to Script Mode in the upper-right corner of the new synchronization node. Select OK to enter the Script Mode.

Note:
Script Mode supports more features, such as synchronous task editing if the network is not up-to-date.

5. Click Import Template in the upper-right corner of the script pattern. Select the data source type for read/write respectively in the pop-up window, and then click OK to generate the initial script.
Configure the synchronization task reader

After creating the synchronization task, the reader basic configurations are generated with the imported template. Now you can manually configure the reader data source and the target table information of the data synchronization task.

```json
{"type": "job", "version": "2.0", "Steps": [] // above is configured for the entire synchronization task header code, do not make modifications. The reader configurations are as follows:
{
    "stepType": "mysql",
    "parameter": {
        "datasource": "MySQL",
        "column": [
            "id",
            "value",
            "table"
        ],
        "socketTimeout": 3600000,
        "connection": [
            {
                "datasource": "MySQL",
                "table": [
                    "`case`
                ]
            }
        ],
        "where": "",
        "splitPk": "",
        "encoding": "UTF-8"
    },
    "name": "Reader",
    "category": "reader" // description classified as reader
read end
}, //The above are reader configurations.
```

Configurations:

- **Type**: Specifies the synchronization task for this submission. Only the job parameter is supported, so you can only enter a job.
- **Version**: The version number currently supported by all jobs is 1.0 or 2.0.

For more information on configuring the read side for specific parameter settings and code descriptions, see the Script Mode section in Configuring reader.

**Note:**

Many tasks require incremental synchronization of data when configuring read data sources, you can now obtain the date in conjunction with what DataWorks provided to complete the requirement #unique_18 to obtain the incremental data.
Configure the synchronization task writer

You can manually configure the writer data source and the target table information for the data synchronization task after configuring the reader data source.

```json
{ //The writer configurations are as follows:
  "stepType": "odps",
  "parameter": {
    "partition": "",
    "truncate": true,
    "compress": false,
    "datasource": "odps_first",
    "column": [
      "*
    ],
    "emptyAsNull": false,
    "table": ""
  },
  "name": "Writer",
  "category": "writer" // instructions are classified as writer
write end
  }
}, //The above are reader configurations.
```

For more information on configuring the write-side information, see the Script Mode section of Configuring writer.

Note:

For most tasks, you need to select a Write mode based on data sources, such as overwrite or append mode. If you have Write control requirements, see Configuring writer to choose the write mode.

Configure mapping

The script mode only supports in-row mapping, that is, the Reader "columns" correspond to the Writer "columns" sequentially from top-to-bottom.

Note:

Check if the field types mapped between the columns are data compatible.

Synchronous task efficiency settings

The efficiency configuration is required when the preceding steps are configured. The Setting domain describes the job configuration parameters in addition to the source, destination, and configuration parameters for task global information. Efficiency can be configured in the setting field, including DMU
setting, synchronization concurrency setting, synchronization rate setting, dirty data setting, and resource group setting.

```
"setting": {
  "errorLimit": {
    "record": "1024" // dirty data entry settings
  },
  "speed": {
    "throttle": false, // do you want to limit the speed?
    "concurrent": 1, // synchronous concurrency number
    "dmu": 1 // DMU quantity settings
  }
},
```

Configurations:

- **DMU**: The billing unit for data integration.

Note:
The configured DMU value limits the maximum concurrency value.

- When you configure **Synchronization Concurrency**, the data records are separated into several tasks based on the specified reader shard key. These tasks run simultaneously to improve the transmission rate.

- **Synchronous rate**: The synchronous rate setting protects the read-side database from fast extraction speed, and reduces pressure on the source library. It is recommended to throttle the synchronization rate and configure the extraction rate properly based on the database source configurations.

- **Dirty data** is set to control the synchronized data quality. It supports setting a threshold for dirty data records. If the number of dirty data records exceeds the threshold during job transmission, the job is aborted with an error. For example, the specified maximum error limit is 1024 records in the preceding configuration. When the job dirty data record number is greater than 1024 during the transfer process, an error is reported during exit.

- You can specify a resource group configuration by clicking configure task resource groups in the upper-right corner of the current page.

When a synchronization task is configured, the resource group in which the task runs is specified. By default, the task runs on the default Resource Group. When the project resource scheduling is tight, you can also expand a resource scheduling by adding a Custom Resource Group. The synchronization task is then specified to run on a Custom Resource Group. For more information on how
to add a Custom Resource Group, see Add task resources. You can set configurations based on the data source network conditions, project scheduling resource conditions, and business importance.

**Note:**
When synchronizing data is inefficient, see Optimizing configuration to optimize your synchronization tasks.

Configure scheduling properties

You can set the synchronization task run cycle, run time, task dependency, and more in the scheduling properties. Because the synchronization task starts the ETL job, there are no upstream nodes. We recommend you use the project root node for the upstream configuration at this point.

After completing the synchronization task configuration, save the node and submit.

**1.6.4 Configure reader plug-in**

**1.6.4.1 Configure DRDS Reader**
The Distributed Relational Database Service (DRDS) Reader plug-in allows you to read data from DRDS. At the underlying implementation level, DRDS Reader connects to a remote DRDS database through JDBC and runs corresponding SQL statements to SELECT data from the DRDS database.

Currently, the DRDS plug-in is only adapted by the MySQL engine. DRDS is a distributed MySQL database, and most of the communication protocols are applicable to MySQL user scenario.

Specifically, DRDS Reader connects to a remote DRDS database through the JDBC connector. The SELECT SQL query statements are generated and sent to the remote DRDS database based on configurations. Then, the run SQL statements and the returned results are assembled into abstract datasets using the custom data types of data synchronization. Datasets are passed to the downstream writer for processing.

DRDS Reader concatenates the table, column, and WHERE information you configured into SQL statements and sends them to the DRDS database. Unlike the MySQL database, as a distributed database DRDS is unable to adapt all MySQL protocols, and does not support complex clauses such as Join.
DRDS Reader supports most MySQL data types. Check whether your data type is supported.

The following are DRDS Reader converted MySQL data types:

<table>
<thead>
<tr>
<th>MySQL data type</th>
<th>DRDS data management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int, tinyint, smallint, mediumint, and bigint</td>
</tr>
<tr>
<td>Floating point</td>
<td>Float, double, decimal</td>
</tr>
<tr>
<td>String</td>
<td>varchar, char, tinytext, text, mediumtext, or longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>date, datetime, timestamp, time, or year</td>
</tr>
<tr>
<td>Boolean</td>
<td>bit or bool</td>
</tr>
<tr>
<td>Binary</td>
<td>tinyblob, mediumblob, blob, longblob, or varbinary</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The table selected for extraction.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The column name set to be synchronized in the configured table. Field information is described with arrays in JSON. [*] Indicates all columns by default.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• Column pruning is supported, which means you can select some columns to export.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Change column order is supported, which means you can export columns in an order different from the schema order of the table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Constant configuration is supported. You must follow the MySQL SQL syntax format, for example [ &quot;id&quot;, &quot;table&quot;, &quot;1&quot;, &quot;bazhen.csy&quot;, &quot;null&quot;, &quot;to_char(a + 1)&quot;, &quot;2.3&quot;, &quot;true&quot; ].</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• id refers to the ordinary column name,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>table</code> is the name of the column containing reserved words,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1 is an integer constant,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 'bazhen.csy' is a string constant,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• null refers to null pointer,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CHARLENGTH(s) is the function expression to calculate the string length,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2.3 is a floating point,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• and true is a boolean value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Column must contain the specified column set to be synchronized and it cannot be blank.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Attribute: where

Filtering condition. DRDS Reader concatenates an SQL command based on the specified column, table, and WHERE conditions and extracts data according to the SQL statement. For example, you can set the WHERE condition during a test. In actual business scenarios, the data on the current day is usually required to be synchronized, in which case you can set the WHERE condition to STRTODATE(‘${bdp.system.bizdate}’, ‘%Y%m%d’) <= todaysdate AND todaysdate < DATEADD(STRTODATE(‘${bdp.system.bizdate}’, ‘%Y%m%d’), interval 1 day).

- The where condition can be effectively used for incremental synchronization.
- If the where condition is not set or is left null, full table data synchronization is applied.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| where     | Filtering condition. DRDS Reader concatenates an SQL command based on the specified column, table, and WHERE conditions and extracts data according to the SQL statement. For example, you can set the WHERE condition during a test. In actual business scenarios, the data on the current day is usually required to be synchronized, in which case you can set the WHERE condition to STRTODATE(‘${bdp.system.bizdate}’, ‘%Y%m%d’) <= todaysdate AND todaysdate < DATEADD(STRTODATE(‘${bdp.system.bizdate}’, ‘%Y%m%d’), interval 1 day).
- The where condition can be effectively used for incremental synchronization.
- If the where condition is not set or is left null, full table data synchronization is applied. | No | N/A |
Development in wizard mode

1. Choose source

Configuration item descriptions:

![Image of configuration items]

Configurations:

- **Data source**: The datasource in the preceding parameter description. Enter the configured data source name.
- **Table**: The table in the preceding parameter description. Select the table for synchronization.
- **Data filtering**: You should synchronize the data filter. Limit keyword filter is not supported yet. SQL syntax's vary with data sources.
- **Splitting key**: You can use a column in the source table as the splitting key. It is recommended to use a primary key or an indexed column as the splitting key. Only integer fields are supported.

During data reading, the data split is based on the configured fields to achieve concurrent reading, improving data synchronization efficiency. The configuration of splitting key is related to the source selection in data synchronization.

**Note:**
The splitting key configuration item is displayed only when you configure the data source.
2. Field mapping: The column in the preceding parameter description.

The Source Table Field on the left maps with the Target Table Field on the right. Click Add Line, and then add a field. Hover the cursor over a line, click Delete, and then delete the line.

- In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.
- Manually edit source table field: Manually edit the fields. Each line indicates a field. The first and end blank lines are ignored.

By clicking Add Row,

- You can enter constants. Each constant must be enclosed in a pair of single quotes, such as 'abc' and '123'.
- Use this function with scheduling parameters, such as ${bizdate}$.
- Enter functions supported by relational databases, such as now() and count(1).
- If the value you entered cannot be parsed, the type is displayed as 'Not Identified'.


3. Channel control

Configurations:

- **DMU**: A unit which measures the resources including CPU, memory, and network bandwidth consumed during data integration. One DMU represents the minimum amount of resources used for a data synchronization task.
- **Concurrent job count**: Maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- **Number of error records**: The maximum number of dirty data records.
- **Task Resource Group**: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend you add a Custom Resource Group, see *Add scheduling resources*.

Development in script mode

**Configure a job to synchronously extract data from an RDBMS database:**

```json
{
    "type": "job",
    "version": "2.0", //Indicates the version.
    "steps": [
        {
            "stepType": "drds", //plug-in name
            "parameter": {
                "datasource": "", //Name of the data source
                "column": [ //column name
                    "id",
                    "name"
                ],
                "where": "", //Filtering condition
                "table": "", //The name of the target table.
                "splitPk": "", //Splitting key
            },
            "Name": "Reader ",
            "category": "reader"
        },
        { //You can locate the corresponding writer plug-in documentation among the following documentations.
            "stepType": "stream", //plug-in name
```
"parameter":{},
"name":"Writer",
"category":"writer"
],
"setting":{
"errorLimit":{
"record":"0"//Number of error records
},
"speed":{
"throttle":false, // do you want to limit the flow?
"concurrent": "1", // Number of concurrency
"DMU": 1 // DMU Value
}
},
"order":{
"hops":[
{
"from":"Reader",
"to":"Writer"
}
]:"Writer"
]
}
}

Additional information

Consistency view

As a distributed database, DRDS cannot provide a consistent view of multiple tables in multiple databases. Unlike MySQL where data is synchronized in a single table of a database, DRDS Reader cannot extract the database or table sharding snapshot from the same time period. That is to say, DRDS Reader obtains different snapshots of table shards when extracting data from different underlying table shards. Therefore, strong consistency cannot be ensured.

Database coding

DRDS provides flexible encoding options, including database-level, table-level, and field-level encoding. Different encodings can also be configured. The priority (from high to low) is field, table, database, and instance. We recommend you use UTF-8 for database encoding at the database level.

DRDS Reader extracts data using JDBC at the underlying level. JDBC is applicable to all types of encodings and can complete transcoding at the underlying level. Therefore, DRDS Reader can identify the encoding and complete transcoding automatically without need to specify the encoding.
DRDS Reader cannot identify inconsistencies between the encoding written to the underlying layer of DRDS and the configured encoding, nor provide a solution. Due to this issue, the exported codes may contain junk codes.

Incremental synchronization

Since DRDS Reader extracts data using JDBC SELECT statements, you can extract incremental data using the SELECT and WHERE conditions with the following methods:

• When database online applications write data into the database, the modify field is filled with the modification timestamp, including addition, update, and deletion (logical deletion). For this type of applications, DRDS Reader only requires the WHERE condition followed by the timestamp of the last synchronization phase.

• For new streamline data, DRDS Reader requires the WHERE condition followed by the maximum auto-increment ID of the last synchronization phase.

In case no field is provided for the business to identify the addition or modification of data, DRDS Reader cannot perform incremental data synchronization and can only perform full data synchronization.

SQL security

DRDS Reader provides query SQL statements for you to SELECT data. DRDS Reader performs no security verification on query SQL. The security during use is ensured by the data synchronization users.

1.6.4.2 Configure HBase reader

The HBase Reader plug-in provides the capability to read data from HBase. At the underlying implementation level, the HBase Reader connects to the remote HBase service with HBase’s Java client. Reads data within the RowKey range specified by Scan, then assemble data into an abstract dataset using custom Data Integration data type, and pass dataset to the downstream Writer for processing.

Supported features

• HBase0.94.x and HBase1.1.x versions

  - If you use HBase 0.94.x, select the HBase094x as the reader plug-in, as follows:

```json
"reader": {
  "plugin": "hbase094x"
}
```
- If you use HBase 1.1.x, select HBase11x as the reader plug-in, as follows:

```
"reader": {
  "plugin": "hbase11x"
}
```

- Normal and multiVersionFixedColumn modes

  - normal mode: Read the latest data version from a HBase table, which is used as an ordinary two-dimensional table (horizontal table). For example:

```java
hbase(main):017:0 is greater than scan 'users'
ROW COLUMN+CELL
lisi column=address:city, timestamp=1457101972764, value=beijing
lisi column=address:country, timestamp=1457102773908, value=china
lisi column=address:province, timestamp=1457101972736, value=beijing
lisi column=info:age, timestamp=1457101972548, value=27
lisi column=info:birthday, timestamp=1457101972604, value=1987-06-17
lisi column=info:company, timestamp=1457101972653, value=baidu
xiaoming column=address:city, timestamp=1457082196082, value=hangzhou
xiaoming column=address:country, timestamp=1457082195729, value=china
xiaoming column=address:province, timestamp=1457082195773, value=zhejiang
xiaoming column=info:age, timestamp=1457082218735, value=29
xiaoming column=info:birthday, timestamp=1457082186830, value=1987-06-17
xiaoming column=info:company, timestamp=1457082189826, value=alibaba
2 row(s) in 0.0580 seconds }
```

The data read from the table is shown as follows:

<table>
<thead>
<tr>
<th>rowKey</th>
<th>address: city</th>
<th>address: country</th>
<th>address: province</th>
<th>info: age</th>
<th>info: birthday</th>
<th>info: company</th>
</tr>
</thead>
<tbody>
<tr>
<td>lisi</td>
<td>beijing</td>
<td>china</td>
<td>beijing</td>
<td>27</td>
<td>1987-06-17</td>
<td>baidu</td>
</tr>
<tr>
<td>xiaoming</td>
<td>hangzhou</td>
<td>china</td>
<td>zhejiang</td>
<td>29</td>
<td>1987-06-17</td>
<td>alibaba</td>
</tr>
</tbody>
</table>

- multiVersionFixedColumn mode: Reads data from a HBase table, which is used as a vertical table. Each record read from the table is shown in the following four columns: rowKey, family:qualifier, timestamp, and value. You must specify the column when reading the data, where each cell value is a record. Multiple records are available if multiple data versions exist, see the following:

```java
hbase(main):018:0 is greater than scan 'users',{VERSIONS=>5}
ROW COLUMN+CELL
lisi column=address:city, timestamp=1457101972764, value=beijing
```
Data read from the table (in four columns):

<table>
<thead>
<tr>
<th>rowKey</th>
<th>Column:qualifier</th>
<th>Timestamp</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lisi</td>
<td>address:city</td>
<td>1457101972764</td>
<td>beijing</td>
</tr>
<tr>
<td>lisi</td>
<td>address:country</td>
<td>1457102773908</td>
<td>china</td>
</tr>
<tr>
<td>lisi</td>
<td>address:province</td>
<td>1457101972736</td>
<td>beijing</td>
</tr>
<tr>
<td>lisi</td>
<td>info:age</td>
<td>1457101972548</td>
<td>27</td>
</tr>
<tr>
<td>lisi</td>
<td>info:birthday</td>
<td>1457101972604</td>
<td>1987-06-17</td>
</tr>
<tr>
<td>lisi</td>
<td>info:company</td>
<td>1457101972653</td>
<td>beijing</td>
</tr>
<tr>
<td>xiaoming</td>
<td>address:city</td>
<td>1457082196082</td>
<td>hangzhou</td>
</tr>
<tr>
<td>xiaoming</td>
<td>address:country</td>
<td>1457082195729</td>
<td>china</td>
</tr>
<tr>
<td>xiaoming</td>
<td>address:province</td>
<td>1457082195773</td>
<td>zhejiang</td>
</tr>
<tr>
<td>xiaoming</td>
<td>info:age</td>
<td>1457082218735</td>
<td>29</td>
</tr>
<tr>
<td>xiaoming</td>
<td>info:age</td>
<td>1457082178630</td>
<td>24</td>
</tr>
<tr>
<td>xiaoming</td>
<td>info:age</td>
<td>1457082186830</td>
<td>1987-06-17</td>
</tr>
<tr>
<td>xiaoming</td>
<td>info:company</td>
<td>1457082189826</td>
<td>alibaba</td>
</tr>
</tbody>
</table>

HBase Reader supports HBase data types and converts HBase data types as follows:

<table>
<thead>
<tr>
<th>Data integration internal types</th>
<th>HBase data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>Int, short, and long</td>
</tr>
<tr>
<td>Double</td>
<td>Float and double</td>
</tr>
<tr>
<td>String</td>
<td>String and binarystring</td>
</tr>
</tbody>
</table>
Data integration internal types | HBase data type
---|---
Date | Date
Boolean | Boolean

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>haveKerberos</td>
<td>If haveKerberos is true, the HBase cluster must use Kerberos for authentication.</td>
<td>No</td>
<td>False</td>
</tr>
</tbody>
</table>

**Note:**
- If the value is true, the following five parameters related to Kerberos authentication must be configured: `kerberosKeytabFilePath`, `kerberosPrincipal`, `hbaseMasterKerberosPrincipal`, `hbaseRegionserverKerberosPrincipal`, and `hbaseRpcProtection`.
- If the HBase cluster is not authenticated with Kerberos, these six parameters are not required.

<table>
<thead>
<tr>
<th>attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbaseConfig</td>
<td>The configuration information provided by each HBase cluster for the Data Integration client connection is stored in the <code>hbase-site.xml</code>. Contact your HBase PE for configuration information, and convert the configuration into JSON format. Multiple HBase client configurations can be added, for example, you can configure the cache and batch scan to optimize the servers interaction.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>mode</td>
<td>Read modes of HBase. The “normal” and “multiVersionFixedColumn” are supported.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The HBase table name to be read and is case sensitive.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>encoding</td>
<td>The encoding method is UTF-8 or GBK. This encoding is used when the HBase byte[] stored in binary form is converted into a String.</td>
<td>No</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The read HBase field. This item is required for both normal and multiVersionFixedColumn modes.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
|           | · In normal mode:<br>The HBase columns specified by “name” for reading must be in the format of column family:column name except for RowKey.  
The “type” specifies the data source type.  
The “format” specifies the date format, and “value” specifies the current type as a constant. The system does not read HBase data, but generates corresponding columns based on “value”. The configuration format is shown as follows:<br>

```
"column":
[
  {
    "name": "rowkey",
    "type": "string",
  },
  {
    "value": "test",
    "type": "string",
  }
]
```

Under normal mode, you must enter the type and select an information from name or value for the specified Column information.<br>
· In multiVersionFixedColumn mode:<br>The HBase columns specified by the item name for reading must be in the format of column family:column name except for RowKey. The constant column is not supported in multiVersionFixedColumn mode. The configuration is as follows:<br>

```
"column":
[
  {
    "Name": "rowkey",
    "type": "string",
  },
  {
    "name": "info: age",
    "value": "test",
    "type": "string",
  }
]
```
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>Specifies the read RowKey range of the HBase reader.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• startRowkey: Specifies the start RowKey.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• endRowkey: Specifies the end RowKey.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• sBinaryRowkey: Specifies the method for converting the configured startRowkey and endRowkey to byte[]. By default, this parameter is false. If the parameter is true, Bytes.toBytesBinary(rowkey) is called for conversion. If the parameter is false, Bytes.toBytes(rowkey) is called. The configuration format is shown as follows.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;range&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;startRowkey&quot;:&quot;aaa&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;endRowkey&quot;:&quot;ccc&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;isBinaryRowkey&quot;:false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scanCacheSize</td>
<td>The number of lines read by the HBase client from the server every time when the RPC is performed.</td>
<td>No</td>
<td>256</td>
</tr>
<tr>
<td>scanBatchSize</td>
<td>The number of columns read by the HBase client from the server every time when the RPC is performed.</td>
<td>No</td>
<td>1,000</td>
</tr>
</tbody>
</table>

Development in wizard mode

**Currently, development in Wizard Mode is not supported.**

Development in script mode

**Configure a job to extract data from HBase to the local machine under normal mode.**

```json
{
  "type":"job",
  "version":"2.0", //Indicates the version.
  "steps":[
    {
      "stepType":"hbase", plug-in name
      "parameter":{
        "mode": "normal", //read HBase mode, supports normal mode, multiVersionFixedColumn Mode
        "scanCacheSize": 256, //Number of lines read by the HBase client from the server every time when RPC is performed.
        "scanBatchSize": 100 " //The number of columns that the HBase client reads per rpc from the server.
    }
  ]
}```
"hbaseVersion": "9.4x/11x", //hbase version
"column":[//Field
    {
        "name":"rowkey", //field name
        "type":"string" //data type
    },
    {
        "name":"columnFamilyName1: columnName1 ",
        "type":"string",
    },
    {
        "name":"columnFamilyName2: columnName2",
        "format":"yyyy-MM-dd",
        "type":"date",
    },
    {
        "name":"columnFamilyName3: columnName3",
        "type":"long"
    }
],
"range":{ //specify the rowkey range that the HBase Reader reads.
    "endRowkey":"", //specify end rowkey.
    "isBinaryRowkey":true,//Specify the method for converting configured startRowkey and endRowkey to byte[]. The default value is false. If it is true, Bytes.toBytesBinary(rowkey) is called for conversion. If it is false, Bytes.toBytes(rowkey) is called.
    "startRowkey":"", //specify the start rowkey.
    "maxVersion":"", //specify the number of versions read by hbase reader in Multi-version Mode
    "encoding":"UTF-8", //encoding format
    "table":"ok", //The name of the target table.
    "hbaseConfig":{ // configuration information required to connect to the hbase cluster, JSON format.
        "hbase.zookeeper.quorum":"hostname",
        "hbase.rootdir":"hdfs://ip:port/database",
        "hbase.cluster.distributed":"true"
    }
},
"name":"Reader",
"category":"reader"
},
{ //The following is a reader template. You can find the corresponding reader plug-in documentations.
    "stepType":"stream",
    "parameter":{},
    "name":"Writer",
    "category":"writer"
}
],
"setting":{
    "errorLimit": {
        "record":"0"//Number of error records
    },
    "speed": {
        "throttle":false, //False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
        "concurrent":"1", //Number of concurrent tasks
        "dmu":1//DMU Value
    }
},
"order":{
1.6.4.3 Configuring HDFS Reader

This topic describes how to configure the HDFS Reader. HDFS Reader provides the ability to read data stored by the distributed file systems. At the underlying implementation level, HDFS Reader retrieves data on the distributed file system, and converts data into a Data Integration transport protocol and transfers it to the Writer.

HDFS Reader provides the ability to read file data from the Hadoop distributed file system HDFS and converts data into a Data Integration transport protocol.

For example:

By default, the TextFile is the storage format for creating Hive tables without data compression. Essentially, the TextFile stores data in HDFS as text, and the HDFS Reader implementation is similar to that of an OSS Reader for Data Integration. ORCFile is the acronym for Optimized Row Columnar File, which is the optimized RCFile. This file format provides an efficient method for storing Hive data. HDFS Reader utilizes the OrcSerde class provided by Hive to read and parse ORCFile data.

Note:

Data synchronization requires an admin account and files read/write permissions.

Usage:
• Create an admin user and home directory to specify a user group and additional
group, and for granting file permissions.

  useradd -m -G supergroup -g hadoop -p admin admin

  - -G supergroup: Specifies the additional group to which the user belongs.
  - -g hadoop: Specifies the user group to which the user belongs.
  - -p admin admin: Add a password to the admin user.

• View the contents of the files in this directory.

  hadoop fs -ls /user/hive/warehouse/hive_p_partner_native

When using Hadoop commands, the format is hadoop fs -command. The
command means command.

• Copies the file part-00000 to the local file system.

  hadoop fs -get /user/hive/warehouse/hive_p_partner_native/part-
  00000

• Edit the file you just copied.

  vim part-00000

• Exits the current user.

  exit

• Connects the host from the list and create an admin account on each attached
host.

  pssh -h /home/hadoop/slave4pssh useradd -m -G supergroup -g hadoop
  -p admin admin

  - pssh -h /home/hadoop/slave4pssh: Connect to the host from the manifest
  file.
  - useradd -m -G supergroup -g hadoop -p admin admin: Create an admin
  account.

Supported functions

Currently, HDFS Reader supports the following features:

• Supports TextFile, ORCFile, rcfile, sequence file, csv, and parquet file formats.
The file logically has a two-dimensional table.
- Supports reading multiple data types represented by Strings and supports column pruning and column constants.
- Supports recursive reading and regular expressions ".*" and ".?".
- Supports ORCFile data compression, and currently supports the SNAPPY and ZLIB compression modes.
- Supports data compression for sequence files, and currently supports the lzo compression mode.
- Supports concurrent reading of multiple files.
- Supports the following compression formats for the csv type: gzip, bz2, zip, lzo, lzo_deflate, and snappy.
- In the current plug in, the Hive version is 1.1.1, and the Hadoop version is 2.7.1 (Apache [is compatible with JDK 1.6]). Data can be written normally in the testing environments of Hadoop 2.5.0, Hadoop 2.6.0, and Hive 1.2.0. For other versions, further tests are required.

**Note:**
Currently, HDFS Reader does not support multi-thread concurrent reading of a single file, which requires internal splitting algorithm of the file.

**Supported data types**

**RCfile**

If the synchronized HDFS file type is a RCfile, you must specify the column data type in the Hive table under “column type” because the data storage mode varies with the data type during the RCfile underlying storage. The HDFS Reader does not support accessing and querying Hive metadata databases. If the column type is BIGINT, DOUBLE, or FLOAT, enter respectively BIGINT, DOUBLE, or FLOAT. If the column type is varchar or char, enter the string for the same purpose.

RCFile data types are converted into default internal types supported by Data Integration, as shown in the following comparison table.

<table>
<thead>
<tr>
<th>Type classification</th>
<th>HDFS data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Tinyint, smallint, int, and bigint</td>
</tr>
<tr>
<td>Float</td>
<td>Float, Double, decimal</td>
</tr>
<tr>
<td>String type</td>
<td>String, Char, and Varchar</td>
</tr>
<tr>
<td>Date and time type</td>
<td>Date and timestamp</td>
</tr>
</tbody>
</table>
### ParquetFile

By default, the ParquetFile data types are converted into internal types supported by Data Integration, as shown in the following comparison table.

<table>
<thead>
<tr>
<th>Type classification</th>
<th>HDFS data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int32, int64, and int96</td>
</tr>
<tr>
<td>Floating point</td>
<td>Float and double</td>
</tr>
<tr>
<td>String type</td>
<td>FIXED_LEN_BYTE_ARRAY</td>
</tr>
<tr>
<td>Date and time type</td>
<td>Date and timestamp</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>BINARY</td>
</tr>
</tbody>
</table>

### TextFile, ORCFile, and SequenceFile

Given that the metadata of TextFile and ORCFile file tables is maintained and stored in the database maintained by Hive, such as MySQL. Currently, HDFS Reader does not support Hive metadata database access and query, so you must specify a data type for conversion.

By default, the TextFile, ORCFile, and SequenceFile data types are converted into internal types supported by Data Integration, as shown in the following comparison table.

<table>
<thead>
<tr>
<th>Category</th>
<th>HDFS data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Tinyint, smallint, int, and bigint</td>
</tr>
<tr>
<td>Floating point</td>
<td>FLOAT and DOUBLE</td>
</tr>
<tr>
<td>String type</td>
<td>String, Char, VARCHAR, Struct, MAP, Array, Union, BINARY</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date and timestamp</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

### Notes:

50

Issue: 20191220
• LONG: Represents an INTEGER string in the HDFS file, such as 123456789.
• DOUBLE: Represents a DOUBLE string in the HDFS file, such as 3.1415.
• BOOLEAN: Represents a BOOLEAN string in the HDFS file, such as true or false and is case-insensitive.
• DATE: Represents a date and time string in the HDFS file, such as 2014-12-31 00:00:00.

Note:
The TIMESTAMP data type supported by Hive can be accurate to the nanosecond, so the TIMESTAMP data content stored in TextFile and ORCFile can be in the format like “2015-08-21 22:40:47.397898389”. If the converted data type is set as the Date for Data Integration, the nanosecond part is truncated after conversion. If you want to retain this part, set the converted data type as the String for Data Integration.
Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| path      | It refers to the read file path. If you want to read multiple files, use a regular expression to match all of them, such as `/hadoop/data_201704*`.  
  - If a single HDFS file is specified, the HDFS Reader only supports single-threaded data extraction.  
  - If multiple HDFS files are specified, the HDFS Reader supports multiple-threaded data extraction, and the number of concurrent threads is determined by the task speed (mbps). The actual number of initiated concurrent threads is the smaller of the number of HDFS files to be read and set task speed.  
  - When the wildcard is specified, the HDFS Reader attempts to traverse multiple files. For example: When the path `"/"` is specified, the HDFS Reader reads all files under the `"/"` directory. When `"/bazhen/"` is specified, the HDFS Reader reads all files under the bazhen directory. Currently, the HDFS Reader only supports wildcards that are asterisks (*) and question marks(?), and the syntax is similar to that of common Linux command wildcards.  
  | Yes       | N/A        |

Note:
The actual number of initiated concurrent threads is the smallest number of HDFS files read and set job speed.

Data Integration considers all files to be read in the same synchronization job as one data table. For this reason, you must ensure all those files adapt the same schema information and grant read permission to Data Integration.

Note on reading partitions: During Hive table creation, you can specify partitions. For example, after creating the partition(day="20150820",hour="09"), two directories with the name of /20150820 and /09 respectively are created in the table catalog of the HDFS file system and /20150820 is the parent directory of /09.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileType</td>
<td>The file type. Currently, only text, orc, rc, seq, csv, or parquet are supported. HDFS Reader can automatically identify files that are ORCFile, RCFile, Sequence File, TextFile, and csv types. Use the appropriate reading policy for the corresponding file type. Before data synchronization, the HDFS Reader checks whether all synchronized file types under the specified path are consistent with the fileType. The synchronization task fails if the synchronized file types are inconsistent to the fileType. The parameter values list that can be configured by fileType is as follows.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- text: The TextFile format.
- orc: The ORCFile format.
- rc: The RCFile format.
- seq: The sequence file format.
- csv: The common HDFS file (logical two-dimensional table) format.
- parquet: The common parquet file format.

**Note:**
Because TextFile and ORCFile are different file formats, the HDFS Reader parses these two file types differently. For this reason, the converted format results varies when converting complex compound types supported by Hive, such as map, array, struct, and union to the String type supported by Data Integration. The following uses map type as an example.

- After being parsed and converted to the String type supported by Data Integration, the ORCFile map type is {job=80, team=60, person=70}.
- After being parsed and converted to the String type supported by Data Integration, the TextFile map type is job:80, team:60, person:70.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| column     | The list of fields read, when the type is the source data. The index indicates the column in which the current column location (starts from 0), and the value indicates the current type is constant and data is not read from the source file, but the corresponding column is automatically generated based on the value. By default, you can read data by taking the String as the only type. The configuration is as: "column": ["*"].<br>The column field can also be configured as follows:<br>[
  {
    "type": "long",
    "index": 0  // Retrieves the int field from the first column of the local file text
  },
  {
    "type": "string",
    "value": "alibaba"  // HDFS Reader internally generates the alibaba string field as the current field
  }
]                                                                 | Yes     | N/A           |
<p>| fieldDelim | It refers to the read field delimiter. The file delimiter is required when the HDFS Reader reads the TextFile data, and by default the delimiter is a comma (,). Field delimiters are not required if none are specified when the HDFS Reader reads the ORCFile data. The Hive default delimiter is \.&lt;br&gt;• To use each row as the target, use characters excluded from the row content as the delimiter, such as the invisible characters \u0001.&lt;br&gt;• Additionally, \n cannot be used as the delimiter. | No      | ,             |
| encoding   | Encoding the read files.                                                                                                                                                                                  | No      | UTF-8         |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullFormat</td>
<td>Text files do not allow defining null (null pointer) with a standard string. Data Integration provides nullFormat to define which strings can be expressed as null. For example, when nullFormat: &quot;null&quot; is configured. If the source data is &quot;null&quot;, it is considered a null field in Data Integration.</td>
</tr>
<tr>
<td>compress</td>
<td>It refers to fileType csv file compression formats, which currently supports gzip, bz2, zip, lzo, lzo_deflate, hadoop-snappy, and framing-snappy.</td>
</tr>
</tbody>
</table>

**Note:**
- Two lzo compression formats are available: lzo and lzo_deflate. Select the corresponding configuration scenario.
- Given that no unified stream format is now available for snappy, Data Integration currently only supports the most common two compression formats provided by Hadoop (hadoop-snappy) and Google recommended format (snappy-framed).
- rc is the format of rcfile.
- No entry is required for the orc file type.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>parquetSchema</td>
<td>This parameter is required for parquet format files. It is used to specify the target file structure, and takes effect only when the fileType is parquet. The format is as follows:</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td><code>message MessageType { Required, data type, column name; ..............................; }</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Notes:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MessageType: Any supported value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Required: Required or Optional. We recommend you use Optional.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Data Type: Parquet files support the following data types: boolean, int32, int64, int96, float, double, binary select binary if the data type is string, and fixed_len_byte_array.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note each configuration row and column, including the last one must end with a semicolon.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Configuration example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>message m {  optional int64 id;  optional int64 date_id;  optional binary datetimestring;  optional int32 dspId;  optional int32 advertiserId;  optional int32 status;  optional int64 bidding_req_num;  optional int64 imp;  optional int64 click_num; }</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csvReaderConfig</td>
<td>Reads the CSV file parameter configurations. It is the Map type. This reading is performed by the CsvReader for reading CSV files and involves many configurations. If there are no configurations, the default values are used.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td><strong>Common configuration:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>csvReaderConfig</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;safetySwitch&quot;: false,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;skipEmptyRecords&quot;: false,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;useTextQualifier&quot;: false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Development in script mode

A script template can be imported for development. The following is a script configuration sample. For relevant parameters, see Parameter Description.

```json
{
    "type": "job",
    "version": "2.0",
    "steps": [
    {
        "stepType": "hdfs", // plug-in name
        "parameter": {
            "path": ", // file path to read
            "datasource": ", //Name of the data source
            "column": [
                {
                    "index": 0, //serial number
                    "type": "string" //Field Type
                },
                {
                    "index": 1,
                    "type": "long"
                },
                {
                    "index": 2,
                    "type": "double",
                },
                {
                    "index": 3,
                    "type": "boolean"
                },
                {
                    "format": "yyyy-MM-dd HH:mm:ss", // time format
                    "index": 4,
                    "type": "date",
                }
            ],
            "fieldDelimiter": "," //Delimiter of each column
            "Encoding": "UTF-8", // encoding format
            "fileType": "// text type"
        },
        "name": "Reader",
        "category": "reader"
    },
    { //The following is a writer template. You can find the corresponding writer plug-in documentations.
        "stepType": "stream",
        "parameter": {},
        "name": "Writer",
        "Category": "Writer"
    }
    ],
    "setting": {
        "errorLimit": {
            "record": ",//Number of error records"
        },
        "speed": {
            "concurrent": "3", //Number of concurrent tasks
            "throttle": false, //False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
            "dmu": 1 // DMU Value
        }
    }
}
```
1.6.4.4 Configure MaxCompute Reader

The MaxCompute Reader plug-in allows you to read data from MaxCompute. For more information about MaxCompute, see *MaxCompute Overview*.

At the underlying implementation level, the MaxCompute Reader plug-in reads data from the MaxCompute system by using a Tunnel based on the source project, table, partition, table fields and other configured information. For common Tunnel commands, see *Tunnel Command Operations*.

MaxCompute Reader can read both partition and non-partition tables, but cannot read virtual views. To read a partition table, you must specify the partition configuration. For example, to read table t0 with a partition configuration of “pt=1, ds=hangzhou”, you must set the value in the configuration. For a non-partition table, the partition configuration is empty. For table fields, you can specify all or some of the columns sequentially, change the column order arrangement, and specify constant fields and partition columns. (A partition column is not a table field).

Supported data types

MaxCompute Reader supports the following data types in MaxCompute.

<table>
<thead>
<tr>
<th>Data type</th>
<th>MaxCompute data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>bigint</td>
</tr>
<tr>
<td>Floating point</td>
<td>double, decimal</td>
</tr>
<tr>
<td>String</td>
<td>string</td>
</tr>
<tr>
<td>Date</td>
<td>Datetime</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in script mode.</td>
</tr>
<tr>
<td>table</td>
<td>The data table name to be read. It is case-insensitive.</td>
</tr>
</tbody>
</table>
| partition | The partition information of the read data. Linux shell wildcards are allowed ("" represents 0 or multiple characters, and "?" represents any character.) For example, a partition table named "test" has four partitions: pt=1/ds=hangzhou, pt=1/ds=shanghai, pt=2/ds=hangzhou, and pt=2/ds=beijing.  
  - To read data from partition pt=1/ds=shanghai, configure it to "partition":"pt=1/ds=shanghai".  
  - To read data from all partitions under pt=1, configure it to "partition:"pt=1/ds=*".  
  - To read data from all partitions of the "test" table, configure it to "partition":"pt=pt/ds=*". | This configuration is required for partition tables, but can be left empty for non-partition tables. | None |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
</table>
| column    | The MaxCompute source table column information. For example, the fields of a table named “test” are id, name, and age.  
  - To read the fields in turn, configure it to "column":"["id","name","age"]" or "column":"[*]".  
  - We do not recommend configuring the extracted field with an asterisk (*) because it indicates every table field is read sequentially. If you change the order or table field types, add or delete some table fields. It is likely the source table columns cannot be aligned with the target table columns, causing errors or even exceptions.  
  - To read name and id sequentially, configure it to: "coulumn":["name","id"].  
  - To add a constant field in the fields extracted from the source table to match the target table field order. For example, if the data values you want to extract are values of age, name, constant date "1988-08-08 08:08:08", and id columns, configure it to: "column":["age","name","'1988-08-08 08:08:08'","id"]", with the constant value enclosed by '. In internal implementation, any field enclosed by ' is considered a constant field, and its value is the content in the ''.  
  - MaxCompute Reader does not use Select SQL statements for extracting table data. Therefore, you cannot specify field functions.  
  - The column must contain the specified synchronized column set and cannot be blank. | Yes | None |
Development in wizard mode

1. Choose source

Configure the synchronization task data source and destination.

- **Data source**: The datasource in the preceding parameter description. Enter the configured data source name.
- **Table**: The table in the preceding parameter description. Select the table for synchronization.

**Note:**
If you specify all columns, you can configure them in the column. For example, "column ": [""]. Partition supports configuration methods that configure multiple partitions and wildcard characters.

- "partition": "pt=20140501/ds=*": Reads data from all partitions in ds.
- "partition": "pt=top?" The question mark (?) means whether the preceding character exists. This configuration specifies the two partitions with pt=top and pt=to.

You can enter partition columns for synchronization, such as partition columns with pt. Example: Assuming that the value of each MaxCompute partition is pt =${bdp.system.bizdate}, add the partition name pt to a source table field, ignore the unrecognized mark if any, and proceed to the next step. To synchronize all partitions, configure the partition value to pt=${*}. To synchronize a certain partition, select a partition time value.
2. Field mapping: The column in the preceding parameter description.

   The Source Table Field on the left maps with the Target Table Field on the right. Click Add Line to add a field. To delete a line, move the mouse cursor over a line and click Delete.

   ![Field mapping](image)

   - In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.
   - Automatic formatting: The fields are automatically sorted by rules.
   - Manually edit source table field: Manually edit fields, where each line indicates a field. The first and end blank lines are ignored.

   By clicking Add Row,

   - Each constant must be enclosed in a pair of single quotes, such as 'abc' and '123'.
   - Use this function with scheduling parameters, such as ${bizdate}.
   - Enter functions supported by relational databases, such as now() and count(1).
   - If the value you entered cannot be parsed, the type is displayed as 'Unidentified'.
3. Control the tunnel

![Configuration settings](image)

**Configurations:**

- **DMU:** A unit that measures resources consumed during data integration, including CPU, memory, and network bandwidth. One DMU represents the minimum amount of resources used for a data synchronization task.
- **Concurrent job count:** Maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. Under wizard mode, configure a concurrency for the specified task on the wizard page.
- **The maximum number of errors indicates the maximum number of dirty data records.**
- **Task Resource Group:** The machine on which the task runs, if there are a large number of tasks, the default Resource Group is used for resource pending. We recommend you add a Custom Resource Group. Currently, only East China 1 and East China 2 supports adding custom resource groups. For more information, see [Add scheduling resources](#).

**Development in script mode**

For more information on how to configure a job for extracting data locally from MaxCompute, see the preceding parameter descriptions for details.

```json
{
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "odps", // plug-in name
            "parameter": {
                "partition": [], // the partition where the read data is located
                "isCompress": false, // do you want to compress?
                "datasource": ", // Data Source
                "column": column information for [ // source table
                "id",
                ],
                "emptyAsNull": true,
            }
        }
    ]
}
```
1.6.4.5 Configure MongoDB Reader

The MongoDB Reader plug-in uses Mongo Client, the Java client of MongoDB, to read data from MongoDB. In the latest version of Mongo, the granularity of the DB lock has been reduced from the DB level to the document level. Combined with the powerful indexing function of MongoDB, it allows a high-performance reading of MongoDB.

Note:

- If you are using ApsaraDB for MongoDB, a root account is provided by default. To ensure security, Data Integration only supports using the relevant account of MongoDB for connection. Avoid using the root account as access account when adding and using the MongoDB data source.
- Query does not support the JS syntax.
MongoDB Reader reads data in parallel from MongoDB by means of Data Integration framework. Based on the specified rules, it partitions the data in MongoDB into multiple data fragments, reads them in parallel using the controlling Job program based on the specified rules, and then converts the data types supported by MongoDB to the ones supported by Data Integration individually.

Type conversion list

MongoDB Reader supports most data types in MongoDB. Check whether your data type is supported before using it.

MongoDB Writer converts the MongoDB data types as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>MongoDB data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>int, long, document.int and document.long</td>
</tr>
<tr>
<td>Double</td>
<td>double and document.double</td>
</tr>
<tr>
<td>String</td>
<td>string, array, document.string, document.array and combine</td>
</tr>
<tr>
<td>Date</td>
<td>date and document.date</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool and document.bool</td>
</tr>
<tr>
<td>Bytes</td>
<td>bytes and document.bytes</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A-</td>
</tr>
<tr>
<td>collection Name</td>
<td>The collection name of MongoDB.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>An array of multiple column names of a document in MongoDB.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• name: Column name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• type: Column type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• splitter: MongoDB supports array, but the CDP framework does not. Therefore, the data items read from MongoDB in an array format are joined into a string using this delimiter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>Used to define the range of returned MongoDB data. For example, if you set it to &quot;query&quot;: &quot;{ operationTime: '{$gte': ISODate('${last_day}T00:00:00.424+0800')}, only the data with an operationTime later than or equal to 00:00 of ${last_day} is returned. ${last_day} is DataWorks scheduling parameter of in the format of $[yyyy-mm-dd]. You can use conditional operators ($gt, $lt, $gte, $lte), logical operators (and, or), and functions (max, min, sum, avg, ISODate) supported by MongoDB as needed. For details, see the query syntax of MongoDB.</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Development in wizard mode

**Currently, development in wizard mode is unavailable.**

Development in script mode

To configure a job to extract data locally from MongoDB, please refer to the above parameter descriptions for details.

```json
{
  "type": "job",
  "version": "2.0",
  "steps": [
    "reader": {
      "plugin": "mongodb",
      "parameter": {
        "datasource": "datasourceName",
        "collectionName": "tag_data",
        "query": "",
        "column": [
          {
            "name": "unique_id",
            "type": "string"
          }
        ]
      }
    }
  ]
}
```


"name": "sid",
"type": "string"
},
{
"name": "user_id",
"type": "string"
},
{
"name": "auction_id",
"type": "string"
},
{
"name": "content_type",
"type": "string"
},
{
"name": "pool_type",
"type": "string"
},
{
"name": "frontcat_id",
"type": "array",
"splitter": ""
},
{
"name": "categoryid",
"type": "array",
"splitter": ""
},
{
"name": "gmt_create",
"type": "string"
},
{
"name": "taglist",
"type": "array",
"splitter": ""
},
{
"name": "property",
"type": "string"
},
{
"name": "scorea",
"type": "int"
},
{
"name": "scoreb",
"type": "int"
},
{
"name": "scorec",
"type": "int"
},
{
"name": "a.b",
"type": "document.int"
},
{
"name": "a.b.c",
"type": "document.array",
"splitter": ""
}
1.6.4.6 Configure DB2 reader

The DB2 Reader plug-in enables data reading from DB2. At the underlying implementation level, the DB2 Reader connects to a remote DB2 database through JDBC and runs corresponding SQL statements to select data from the DB2 database.

Specifically, DB2 Reader connects to a remote DB2 database through the JDBC connector. The SELECT SQL query statements are generated and sent to the remote DB2 database based on your configurations. Then, the SQL statements are run and the returned results are assembled into abstract datasets using the custom data types of data integration. Datasets are passed to the downstream writer for processing.

- DB2 Reader concatenates the configured table, column, and WHERE information into SQL statements and sends them to the DB2 database.
- DB2 Reader directly sends configured query SQL information to the DB2 database.

DB2 Reader supports most DB2 data types. Check whether the data type is supported.

DB2 Reader converts DB2 data types as follows:
<table>
<thead>
<tr>
<th>Type classification</th>
<th>DB2 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>Floating point</td>
<td>decimal, real, or double</td>
</tr>
<tr>
<td>String</td>
<td>char, character, varchar, graphic, vargraphic, long varchar, clob, long vargraphic, or dbclob</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date, time, and timestamp</td>
</tr>
<tr>
<td>Boolean</td>
<td>—</td>
</tr>
<tr>
<td>Binary</td>
<td>blob</td>
</tr>
</tbody>
</table>

### Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>jdbcUrl</td>
<td>Information of the JDBC connection to the DB2 database. In accordance, with the DB2 official specification, jdbcUrl in the DB2 format is jdbc:db2://ip:port/database, and you can enter the connection accessory control information.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>username</td>
<td>User name for the data source.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>password</td>
<td>Password corresponding to the specified data source user name.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The table you select for synchronization. Each operation only supports one table synchronization.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The configured table requires a collection of column names synchronized with a JSON array to describe the field information. By default, all column configurations, such as [&quot;*&quot;] are used.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>- Column pruning is supported, which means you can select columns for export.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Changing column order is supported, which means the column export order can be different from the table schema order.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|           | - Constant configuration is supported. You must follow the DB2 SQL syntax format. For example:
|           |   
|           |   - where id refers to the ordinary column name.
|           |   - 1 is an integer numeric constant
|           |   - 'const name' is a String constant (requires a pair of single quotes)
|           |   - null is a null pointer
|           |   - upper('abc_lower') is a function expression
|           |   - 2.3 is a floating point number
|           |   - True is a Boolean Value
<p>|           | - The column must contain the specified column set for synchronization and it cannot be blank. |        |               |
| SplitPk   | If you specify the SplitPk when using the RDBMSReader to extract data, it means fields represented by SplitPk are used for data sharding. Then DataX starts concurrent tasks to synchronize data, which greatly improves the data synchronization efficiency. | No | Null |
|           | - We recommend you use the table primary keys for SplitPk because the primary keys are generally even and less likely to generate data hot spots during data sharding. |        |               |
|           | - Currently, SplitPk only supports data sharding for integer data types. Other types such as floating point, string, and date are not supported. If you specify an unsupported data type, the DB2 Reader reports an error. |        |               |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE</td>
<td>A filtering condition. The DB2 Reader concatenates an SQL command based on the specified column, table, and WHERE clauses. It extracts data according to the SQL statement. In business scenarios, data from the current day are usually required for synchronization. You can specify the where condition as gmt_create &gt; $bizdate. The WHERE clauses can be used to synchronize incremental business data effectively. If the value is null, it will synchronize all information in the table.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>QuerySQL</td>
<td>In some business scenarios, the WHERE clause is insufficient for filtering. In this case, you can customize a filter SQL using QuerySQL. When QuerySQL is configured, the data synchronization system filters data with QuerySQL instead of other configuration items, such as tables and columns. For example, data synchronization after multi-table join, can use <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. When query SQL is configured, DB2 Reader ignores table, column, and WHERE clause configurations.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Fetchsize</td>
<td>Defines the batch data pieces that the plug-in and database servers can fetch each time. The value determines the number of network interactions between the data synchronization system and the server, which greatly improves data extraction performance.</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>

Note:
A value greater than 2048 may cause out-of-memory (OOM) during data synchronization.

Development in wizard mode

Currently, development in wizard mode is unavailable.

Development in script mode

Configure a job to synchronously extract data from a DB2 database:

```
Active/standby synchronous data recovery problem

Active/standby synchronization means that DB2 uses an active/standby disaster recovery mode in which the standby database continuously restores data from the active database through binlog. Because of time differences in active/standby data synchronization, especially in situations, such as network latency. The restored
data in the standby database after synchronization are significantly different from the active database data. That is to say, the data synchronized in the standby database is not a full image of the current active database.

Consistency limits

In data storage, DB2 is a RDBMS system that can provide strong data consistency APIs for querying. For example, if another user writes data to the database during a synchronization task, DB2 Reader does not obtain the newly written data because of the database snapshot features. For the databases snapshot features, see MVCC Wikipedia.

The following are data synchronization consistency features in the single-threaded model of the DB2 Reader. Robust data consistency cannot be guaranteed because DB2 Reader uses concurrent data extraction based on configured information. After DB2 Reader completes data sharding based on SplitPk, multiple concurrent tasks are successively enabled to synchronize data. Because multiple concurrent tasks belong to different read transactions, time intervals exist between concurrent tasks. As a result, the data is incomplete and the data snapshot information is inconsistent.

Currently, consistency snapshot demands in multi-threaded model can only be solved from an engineering perspective. The engineering approaches has both advantages and disadvantages. The following are suggested solutions:

- Use single-threaded synchronization without data sharding. This is slow but can ensure robust data consistency.
- Disable other data writers to ensure the current data is static. For example, you can lock the table or disable standby database synchronization. Note: Disabling the data writer may affect your online business.

Database encoding

The DB2 Reader extracts data using JDBC at the underlying level. JDBC is applicable to all encoding types and can complete transcoding at the underlying level. Therefore, DB2 Reader can identify the encoding and automatically complete transcoding without specifying the encoding.

Incremental synchronization
Since Oracle Reader extracts data using JDBC SELECT statements, you can extract incremental data using SELECT...WHERE... statement in either of the following ways:

- When online database applications write data into the database, the modify field enters the modification timestamp, including addition, update, and deletion (logical deletion). For this type of application, DB2 Reader only requires the WHERE condition followed by the timestamp of the last synchronization phase.
- For new streamline data, DB2 Reader requires the WHERE statement followed by the maximum auto-increment ID of the last synchronization phase.

In the case that no fields are provided for the business to identify added or modified data, the DB2 Reader cannot perform incremental data synchronization and can only perform full data synchronization.

SQL security

The DB2 Reader provides query SQL statements for you to SELECT data. The DB2 Reader does not perform security verification on query SQL.

### 1.6.4.7 Configure MySQL Reader

This topic describes how to configure a MySQL Reader. The MySQL Reader connects to a remote MySQL database through the JDBC connector. The SQL query statements are generated and sent to the remote MySQL database based on your configuration. Then, the SQL statements are run and the returned results are assembled into abstract datasets using the custom data types of data synchronization. Datasets are then passed to the downstream writer for processing.

In short, MySQL Reader reads data from the MySQL database underlying level by using the JDBC connector to connect the MySQL Reader to the remote MySQL database, and runs SQL statements to select data from the MySQL database.

MySQL Reader supports table and view reading. In the table field, you can specify all columns in sequence, specify certain columns, adjust column order, specify constant fields, and configure MySQL functions, such as now().

MySQL Reader supports the following MySQL data types.

<table>
<thead>
<tr>
<th>Type classification</th>
<th>MySQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>int, tinyint, smallint, mediumint, int, bigint</td>
</tr>
<tr>
<td>Floating point</td>
<td>float, double, decimal</td>
</tr>
<tr>
<td>Type classification</td>
<td>MySQL data type</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>String</td>
<td>varchar, char, tinytext, text, mediumtext, longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>date, datetime, timestamp, time, year</td>
</tr>
<tr>
<td>Boolean</td>
<td>bit, bool</td>
</tr>
<tr>
<td>Binary</td>
<td>tinyblob, mediumblob, blob, longblob, varbinary</td>
</tr>
</tbody>
</table>

**Note:**
- Only the field types listed in the preceding table are supported.
- MySQL Reader classifies tinyint(1) as the integer type.

**Type conversion list**

MySQL Writer converts the MySQL data types as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>MySQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int, Tinyint, Smallint, Mediumint, Bigint</td>
</tr>
<tr>
<td>Float</td>
<td>Float, Double, Decimal</td>
</tr>
<tr>
<td>String type</td>
<td>Varchar, Char, Tinytext, Text, Mediumtext, LongText</td>
</tr>
<tr>
<td>Date and time type</td>
<td>Date, Datetime, Timestamp, Time, Year</td>
</tr>
<tr>
<td>boolean</td>
<td>Bool</td>
</tr>
<tr>
<td>Binary</td>
<td>Tinyblob, Mediumblob, Blob, LongBlob, Varbinary</td>
</tr>
</tbody>
</table>

**Parameter description**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>You select a table name that requires synchronization, and a data integration Job can only synchronize one table.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The column name set to be synchronized in the configured table. Field information is described with JSON arrays. [ * ] indicates all columns by default.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• Column pruning is supported, which means you can select some columns to export.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Change of column order is supported, which means you can export the columns in an order different from the schema order of the table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Constant configuration is supported. You must follow the MySQL SQL syntax format, for example [&quot;id&quot;, &quot;table&quot;, &quot;1&quot;, &quot;'mingya.wmy'&quot;, &quot;'null'&quot;, &quot;to_char(a + 1)&quot;, &quot;2.3&quot;, &quot;true&quot;].</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- ID is a normal column name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Table is a column name that contains Reserved Words</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 1 for plastic digital Constants</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 'mingya. wmy' is a String constant (note that a pair of single quotes is required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Null is a null pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- CHAR_LENGTH(s) is the computed String Length Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 2.3 is a floating point number</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- true is a Boolean Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The column must contain the specified column set for synchronization and it cannot be blank.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| SplitPk   | If SplitPk is specified when using MySQL Reader to extract data, it means the fields are represented by SplitPk for data sharding. Data synchronization starts using concurrent tasks to synchronize data, which greatly improves data synchronization efficiency.  
  • We recommend you use the table primary keys for SplitPk because the primary keys are usually even and less likely to generate data hot spots during data sharding.  
  • Currently, SplitPk only supports data sharding for integer data types. Other types such as string, floating point, and date are not supported. If you specify an unsupported data type, the SplitPk is ignored and the data is synchronized using a single channel.  
  • If the SplitPk is unspecified the table data is synchronized using a single channel. For example, when SplitPk is not provided or when the SplitPk value is null. | No | N/A |
| WHERE     | In actual business scenarios, the current day data is usually required for synchronization. You can specify the WHERE clause as gmt_create > $bizdate.  
  • The WHERE clause can be effectively used for incremental synchronization. Full synchronization is performed when the WHERE clause is not specified, for example, when the WHERE key or value is not provided.  
  • You cannot specify limit 10 as the WHERE clause, because it does not conform to MySQL WHERE clause requirements. | No | N/A |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>querySQL (only available in advanced mode)</td>
<td>querySQL is used for customizing a filter SQL in business scenarios, where the WHERE clause is an insufficient filter. When this item is configured, the data synchronization system filters data with this configuration item directly, instead of configuration items, such as tables and columns. For example, for data synchronization after multi-table join, use <code>select a, b from table_a join table_b on table_a.id = table_b.id</code>. When querySQL is configured, MySQL Reader directly ignores the configuration of table, column, WHERE, and SplitPk conditions. The querySQL priority is higher than the table, column, WHERE, and SplitPk. The datasource uses querySQL to parse information, such as a user name and password.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>singleOrMulti (applies only to hardedsharded tables and sharded databases)</td>
<td>Represents a sharded table or sharded databases, and the wizard mode is converted into Script Mode to actively generate this configuration &quot;singleOrMulti&quot;: &quot;multi&quot;. This configuration is not automatically generated by the script task template, and must be added manually, or only the first data source is recognized. singleOrMulti is just the frontend, and the back-end does not use this for sharded table judgment.</td>
<td>Yes</td>
<td>multi</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

Configure the data source and destination of the synchronization task.

![Connections](image)

Configurations:

- **Data source**: The data source in the preceding parameter description. Enter the configured data source name.
- **Table**: The table in the preceding parameter description. Select the table for synchronization.
- **Data filtering**: The data synchronization filtering criteria. Currently, keyword filtering limits are not supported. The SQL syntax is consistent with the selected data source.
- **Shard keys**: You can use a column in the source data table as a shard key. We recommend that you use a primary key or an indexed column as the shard key, and only Integer type fields are supported.

The data shard is based on configured fields during data reading to achieve concurrent reading, and improve data synchronization efficiency.

**Note:**

The shard key configuration is related to the source selection in data synchronization. The shard key configuration item is displayed only when you configure the data source.
2. The field mapping is the column in the above parameter description.

The source table field on the left and the target table field on the right are one-to-one correspondence, click Add row to add a single field and click Delete to delete the current field.

- Peer mapping: Click peer mapping to establish a corresponding mapping relationship in the peer, and take special note of the data type match.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.
- Manually edit source table field: Manually edit fields, where each line indicates a field. The first and end blank lines are ignored.

The function of adding a row is as follows:

- You can enter constants. The value must be enclosed by a pair of single quotes, such as 'abc' and '123'.
- Use this function with scheduling parameters, such as ${bizdate}.
- You can enter functions supported by relational databases, such as now() and count(1).
- If the value entered cannot be parsed, the type is displayed as unidentified.
3. Control the tunnel

**Configurations:**

- **DMU:** A unit which measures the resources including CPU, memory, and network bandwidth consumed during data integration. One DMU represents the minimum amount of resources used for a data synchronization task.
- **Concurrent job count:** The maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- **The maximum number of errors indicates the maximum number of dirty data records.**
- **Task Resource Group:** The machine on which the task runs, if the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend that you add a Custom Resource Group. For more information, see *Add scheduling resources.*

**Development in script mode**

A script sample for a single-library and single-table, for example, can be found in the above parameter descriptions.

```json
{
    "type": "job",
    "version": "1.0"} //Indicates the version.
"steps": [  
{"stepType": "mysql", // plug-in name
"parameter": [{
    "Column": ["id"],
},
"connection": [
    {"querysql": "select a,b from join1 c join join2 d on c.id = d.id;"},
"datasource": ", // Data Source
"table": ["xxx"]
]  
}
```
1.6.4.8 Configure Oracle Reader

This topic describes how to configure an Oracle Reader. The Oracle Reader plug-in provides the capability to read data from Oracle. At the underlying implementation level, Oracle Reader connects to a remote Oracle database through JDBC and runs SELECT statements to extract data from the database.

On the public cloud, RDS or DRDS does not provide the Oracle storage engine. Currently, Oracle Reader is mainly used for private cloud data migration and Data Integration projects.

In short, Oracle Reader connects to a remote Oracle database through the JDBC connector. The SELECT SQL query statements are generated and sent to the remote Oracle database based on your configuration. Then, the SQL statements are run and returned results are assembled into abstract datasets using the data
synchronization custom data types. Datasets are passed to the downstream writer for processing.

- Oracle Reader concatenates configured table, column, and WHERE information into SQL statements and sends them to the Oracle database.
- Oracle sends the querySQL information you configured to the Oracle database.

Type conversion list

Oracle Reader supports most data types in DB2. Check whether your data type is supported.

Oracle Reader converts Oracle data types as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>Oracle data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Number, rawd, integer, Int, and smallint</td>
</tr>
<tr>
<td>Float</td>
<td>Numeric, decimal, float, double precision, real</td>
</tr>
<tr>
<td>String type</td>
<td>Long, Char, NChar, Varchar, Varchar2, NVar2, Clob, NClob, character, character varying, char varying, national character, National char, National Character varying, national char varying and nchar varying</td>
</tr>
<tr>
<td>Date and time type</td>
<td>Timestamp and Date</td>
</tr>
<tr>
<td>Boolean</td>
<td>Bit and Bool</td>
</tr>
<tr>
<td>Binary</td>
<td>Blob, BFile, Raw, and Long Raw</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The name of the selected table that needs to be synchronized.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>column</td>
<td>The column name set to be synchronized in the configured table. Field information is described with JSON arrays. [&quot;***&quot;] indicates all columns by default.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• Column pruning is supported, which means you can select export columns.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Change column order is supported, which means you can export columns in an order different from the table schema order.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Constant configuration is supported, and you need to configure in JSON format.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[&quot;id&quot;, &quot;1&quot;, &quot;'mingya.wmy'&quot;, &quot;null&quot;, &quot;to_char(a + 1)&quot;, &quot;2.3&quot;, &quot;true&quot;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- ID is normal column name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 1 is an integer numeric constant</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 'Mingya.wmy' is a String constant (note that a pair of single quotes is required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Null is a null pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- to_char(a + 1) is an expression</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 2.3 is a floating point number</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- True is a Boolean Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Column is required and cannot be blank.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SplitPk</td>
<td>If you specify the SplitPk when using RDBMSReader to extract data, it means the fields are represented by SplitPk for data sharding. Then, the DataX starts concurrent tasks to synchronize data, which greatly improves data synchronization efficiency.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• If you are using SplitPk, we recommend that you use table primary keys because the primary keys are generally even and less likely to generate data hot spots during data sharding.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The data types supported by SplitPk include the integer, string, floating point, and date.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If SplitPk is left blank, it indicates that no table sharding is required and Oracle Reader synchronizes full data through a single channel.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| WHERE            | The filtering condition. Oracle Reader concatenates an SQL command based on specified column, table, and WHERE clauses and extracts data according to the SQL command. For example, you can set the WHERE clauses as `row_number()` during a test. In actual service scenarios, the incremental synchronization typically synchronizes data generated on the current day. You can specify the WHERE clauses as `id > 2 and sex = 1`.  
   · The WHERE clauses can be effectively used for incremental synchronization.  
   · The WHERE clauses can be effectively used for incremental synchronization. | No      | N/A           |
| querySQL         | In some service scenarios, the WHERE clauses is insufficient for filtering. In such cases, you can customize a SQL filter using this parameter. When this item is configured, the data synchronization system filters data using this configuration item directly instead of configuration items, such as table and column. For example, data synchronization after multi-table join, uses `select a,b from table_a join table_b on table_a.id = table_b.id`. When querySQL is configured, Oracle Reader directly ignores the configuration of tables, columns, and WHERE clauses. | No      | N/A           |
| fetchSize        | It defines the pieces of batch data that the plug-in and database server can fetch each time. The value determines the number of network interactions between the DataX system and the server, which can greatly improve data extraction performance.  
   Note:  
The fetchsize value (> 2048) may cause out of memory (OOM) during the data synchronization process. | No      | 1,024         |
Development in wizard mode

1. Choose source

Configure the source and destination of the synchronization task data.

![Connection Configuration](image)

Configurations:

- **Data source**: The datasource in the preceding parameter description. Enter the data source name configured.
- **Table**: The table in the preceding parameter description. Select the table for synchronization.
- **Data filtering**: You are about to synchronize the data filtering criteria, and limit keyword filtering is not supported for the time being. The SQL syntax is consistent with the selected data source.
- **Shard key**: You can use a column in the source data table as a shard key, it is recommended you use a primary key or an indexed column as a shard key, and that only Integer type fields are supported.

The read data is sharded based on the configured fields to achieve concurrent reading, and improve data synchronization efficiency.

**Note:**

The shard key configuration is related to the source selection in data synchronization. The shard key configuration item is displayed only when you configure the data source.
2. The field mapping is the column in the above parameter description. The source table field on the left and the target table field on the right are one-to-one correspondence, click Add row to add a single field and click Delete to delete the current field.

- Peer mapping: Click Enable Same-Line Mapping to establish a corresponding mapping relationship in the peer, note the data type match.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.
- Manually edit source table field: Manually edit fields, where each line indicates a field. The first and end blank lines are ignored.

The function of adding a row is as follows:

- You can enter constants. The value must be enclosed by a pair of single quotes, such as 'abc' and '123'.
- Use this function with scheduling parameters, such as `${bizdate}`.
- You can enter functions supported by relational databases, such as `now()` and `count(1)`.
- If the value you entered cannot be parsed, the type is displayed as not identified.
3. Control the tunnel

![Channel]

Configurations:

- **Concurrent job count**: Maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- **The maximum number of errors** indicates the maximum number of dirty data records.
- **Task Resource Group**: The machine on which the task runs, if the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend that you add a Custom Resource Group. For more information, see #unique_12.

Development in script mode

Configure a job to synchronously extract data from an Oracle database:

```json
{
   "type": "job",
   "version": "2.0"} //Indicates the version.
"steps": [
   {
      "stepType": "oracle",
      "parameter": {
         "fetchSize": 1024, // The configuration item defines the number of plug-ins and database server-side data acquisition lines per volume
         "datasource": "",
         // fill in the added Data Source Name
         "column": [ // column name
            "id",
            "name"
         ],
         "where": "", //Filtering condition
         "splitPk": "", // cut key
         "table": "" // table name
      },
      "name": "Reader ",
      "category": "reader"
   }
}
```
Additional instructions

Active/standby synchronous data recovery problem

Active/standby synchronization means that Oracle uses an active/standby disaster recovery mode in which the standby database continuously restores data from the active database through binlog. Because of time difference in active/standby data synchronization, especially in situations such as network latency, the restored data in the standby database after synchronization is significantly different from the data of the active database. That is to say, the data synchronized in the standby database currently are not a full image of the active database.

Consistency limits

Oracle is an RDBMS system in terms of data storage, which can provide APIs for strong consistency data querying. For example, if another user writes data to the database during a synchronization task, Oracle Reader does not obtain the newly
written data because of the database snapshot features. For more information on the database snapshot features, see MVCC Wikipedia.

The preceding are characteristics of data synchronization consistency under the Oracle reader single-threaded model, since Oracle reader can use Concurrent Data Extraction based on your configuration information, data consistency is not strictly guaranteed. When the Oracle reader shards are based on the SplitPk data, multiple concurrent tasks are initiated to complete the data synchronization. Since multiple concurrent tasks do not belong to the same read transaction and time intervals exist between the concurrent tasks, the data is incomplete and data snapshot information is inconsistent.

Multi-thread consistent snapshot requirements can only be solved from an engineering perspective. The following are suggested engineering solutions, and you can choose according to your circumstances.

- Use single-threaded synchronization without data sharding. This is slow but can ensure robust data consistency.
- Close other data writers to ensure the current data is static. For example, you can lock the table or close standby database synchronization. The disadvantage is it may affect online businesses.

Database coding problem

The Oracle Reader extracts data using JDBC at the underlying level. JDBC is applicable to all types of encodings and can complete transcoding at the underlying level. Therefore, the Oracle Reader can obtain the encoding and complete transcoding automatically without the need to specify the encoding.

The Oracle Reader cannot identify inconsistencies between the encoding written in the underlying layer of the Oracle system and the configured encoding, nor provides a solution. Due to this issue, **the exported codes may contain junk codes**.

Incremental synchronization

Since Oracle Reader extracts data using JDBC SELECT statements, you can extract incremental data using the SELECT and WHERE clauses using either of the following methods:
When online database applications write data into the database, the modify field is entered with the modification timestamp, including addition, update, and deletion (logical deletion). For this type of applications, Oracle Reader only requires the WHERE clauses followed by the last synchronization phase timestamp.

For new streamline data, the Oracle Reader requires the WHERE clauses followed by the maximum auto-increment ID of the last synchronization phase.

In case no field is provided for the business to identify added or modified data, the Oracle Reader cannot perform incremental data synchronization and can only perform full data synchronization.

SQL security

The Oracle Reader provides querySQL statements for you to SELECT data. The Oracle Reader does not perform security verification on querySQL.

1.6.4.9 Configure OSS Reader

The OSS Reader plug-in provides the ability to read data from OSS data storage. In terms of underlying implementation, OSS Reader acquires the OSS data using official OSS Java SDK, converts the data to the data synchronization protocol, and passes it to Writer.

- If you want to learn more about OSS products, see the OSS product overview.
- For details about OSS Java SDKs, see Alibaba Cloud OSS Java SDK.
- For details on processing non-structured data such as the OSS data, see Process non-structured data.

The OSS Reader provides the capability to read data from a remote OSS file and convert data to the Data Integration and datax protocol. OSS file itself is a non-structured data storage. For Data Integration and datax, OSS Reader currently supports the following features:

- Only supports reading TXT files and the schema in the TXT file must be a two-dimensional table.
- Supports CSV-like format files with custom delimiters.
- Supports reading multiple types of data (represented by String) and supports column pruning and column constants.
- Supports recursive reading and filtering by File Name.
• Supports text compression. The available compression formats include gzip, bzip2, and zip.

Note:
Multiple files cannot be compressed into one package.

• Supports concurrent reading of multiple objects.

The following are not supported currently:

• Multi-thread concurrent reading of a single object (file).
• Technically, the multi-thread concurrent reading of a single compressed object is not supported.

OSS Reader supports the following data types of OSS: BIGINT, DOUBLE, STRING, DATATIME, and BOOLEAN.

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| Object    | The object information for the OSS, where you can support filling in multiple objects. For example, if the bucket of xxx contains a yunshi folder that has ll.txt file, the object is directly specified as yunshi/ll.txt.  
  - If a single OSS object is specified, OSS Reader only supports single-threaded data extraction. We are planning to provide the function to concurrently read a single non-compressed object with multiple threads.  
  - If multiple OSS objects are specified, OSS Reader can extract data with multiple threads. The number of concurrent threads is specified based on the number of channels.  
  - If a wildcard is specified, OSS Reader attempts to traverse multiple objects. For details, see OSS product overview. | Yes | N/A |

Note:
Data synchronization system identifies all objects synchronized in a job as a same data table. You must ensure that all objects are applicable to the same schema information.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>It refers to the list of fields read, where the type indicates the source data type. The index indicates the column in which the current column locates (starts from 0), and the value indicates the current type is constant. The data is not read from the source file, but the corresponding column is automatically generated according to the value. By default, you can read data by taking the String as the only type. The configuration is as follows:</td>
<td>Yes</td>
<td>Read all according to string type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fieldDelimiter</td>
<td>The read field separator.</td>
<td>Yes</td>
<td>,</td>
</tr>
<tr>
<td>compress</td>
<td>The compression file type. It is left empty by default, which means no compression is performed. Supports the following compression types: gzip, bzip2, and zip.</td>
<td>No</td>
<td>Do not compress</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>encoding</td>
<td>Encoding of the read files.</td>
<td>No</td>
<td>UTF-8</td>
</tr>
<tr>
<td>nullFormat</td>
<td>Defining null (null pointer) with a standard string is not allowed in text files. Data synchronization system provides nullFormat to define which strings can be expressed as null. For example, if the source data is &quot;null&quot;, if you configure the nullformat = &quot;null &quot;, the data synchronization system is treated as a null field.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Skipheader</td>
<td>The header of a file in CSV-like format is skipped if it is a title. Headers are not skipped by default. skipHeader is not supported for file compression.</td>
<td>No</td>
<td>false</td>
</tr>
<tr>
<td>csvReaderConfig</td>
<td>Reads the parameter configurations of CSV files. It is the Map type. This reading is performed by the CsvReader for reading CSV files and involves many configuration items, whose defaults are used if they are not configured.</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. **Choose source**

Configure the source and destination of the data for the synchronization task.

![Source and Destination Configuration](image)

**Configurations:**

- **Data source:** The datasource in the preceding parameter description. Enter the data source name you configured.
- **Object prefix:** Object in the preceding parameter description.

**Note:**
If your OSS file name has a section named according to the time of day, such as aaa/20171024abc.txt, about the object system parameters, `aaa/${bdp.system.bizdate}abc.txt` can be set.

- **Column delimiter:** `fieldDelimiter` in the preceding parameter description, which defaults to ",".
- **Encoding format:** `Encoding` in the preceding parameter description, which defaults to utf-8.
- **null value:** `nullFormat` in the preceding parameter description. Enter the field to be expressed as null into a text box. If source end exists, the corresponding field is converted to null.
- **Compression format:** `Compress` in the preceding parameter description, which defaults to "no compression".
- **Whether to include the table header:** `skipHeader` in the preceding parameter description, which defaults to "No".

2. The field mapping which is the column in the above parameter description.

The source table field on the left and the target table field on the right are one-to-one correspondence, click Add row to add a single field and click Delete to delete the current field.

- **Peer mapping:** Click Enable Same-Line Mapping to establish a corresponding mapping relationship in the peer, note that match the data type.
- **Manually edit source table field:** Manually edit the fields. Each line indicates a field. The first and end blank lines are ignored.
3. Control the tunnel

![Image of Control the tunnel](image)

Configurations:

- DMU: A unit which measures the resources, including CPU, memory, and network bandwidth consumed during data integration. One DMU represents the minimum amount of resources used for a data synchronization task.
- Concurrent job count: Maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- The maximum number of errors indicates the maximum number of dirty data records.

Development in script mode

The following is a script configuration sample. For details about parameters, see the preceding parameter description:

```json
{
    "type": "job",
    "version": "2.0", // Indicates the version.
    "steps": [
        {
            "stepType": "oss", // plug-in name
            "parameter": {
                "nullFormat": "", // nullformat defines which strings can be expressed as null?
                "compress": "", // text compression type
                "datasource": "", // Data Source
                "column": [ // Field
                    { "index": 0, // column sequence number
                      "type": "string" // data type
                    },
                    { "index": 1,
                      "type": "long"
                    },
                    { "index": 2,
                      "type": "double"
                    }
                ]
            }
        }
    ]
}
```
### 1.6.4.10 Configuring FTP Reader

FTP Reader provides the capability to read data from a remote FTP file system. At the underlying implementation level, FTP Reader acquires the remote FTP file data, converts data to the data synchronization and transmission protocol, and transmits it to Writer.

What is saved to the local file is a two-dimensional table in a logic sense, for example, text information in a CSV format.
FTP Reader allows you to read data from a remote FTP file and convert the data to the data synchronization protocol. Remote FTP file itself is a non-structured data storage file. For data synchronization, FTP Reader currently supports the following features:

- Only supports reading TXT files and the schema in the TXT file must be a two-dimensional table.
- Supports CSV-like format files with custom delimiters.
- Supports reading multiple types of data (represented by String) and supports column pruning and column constants.
- Supports recursive reading and filtering by File Name.
- Supports text compression. The available compression formats, include gzip, bzip2, zip, lzo, and lzo_deflate.
- Supports concurrent reading of multiple files.

The following two features are not supported currently:

- Multi-thread concurrent reading of a single file. This feature involves the internal splitting algorithm of a single file (under planning).
- Technically, the multi-thread concurrent reading of a single compressed file is not supported.

The remote FTP file itself does not provide data types, which are defined by DataX FtpReader:

<table>
<thead>
<tr>
<th>Internal DataX type</th>
<th>Data type of a remote FTP file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>Long</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
</tr>
</tbody>
</table>
## Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| path      | The path of the remote FTP file system. Multiple paths can be specified.  
- If a single remote FTP file is specified, FTP Reader only supports single-threaded data extraction. We are planning to provide the function to concurrently read a single non-compressed file with multiple threads.  
- If multiple remote FTP files are specified, FTP Reader can extract data with multiple threads. The number of concurrent threads is specified based on the number of channels.  
- If a wildcard is specified, FTP Reader attempts to traverse multiple files. For example, when / is specified, FTP Reader reads all the files under the / directory. When /bazhen/ is specified, FTP Reader reads all the files under the bazhen directory. Currently, FTP Reader only supports * as the file wildcard. | Yes- | N/A |

### Note:

- The data synchronization system identifies all text files synchronized in a job as a same data table. You must ensure that all files are applicable to the same schema information.
- You must ensure that the file to be read is in CSV-like format, and the read permission must be granted to the data synchronization system.
- If no matching file exists for extraction in the path specified by Path, an error may occur in the synchronization task.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>It refers to the list of fields read, where the type indicates the type of source data. The index indicates the column in which the current column locates (starts from 0), and the value indicates that the current type is constant. The data is not read from the source file, but the corresponding column is automatically generated according to the value. By default, you can read data by taking String as the only type. The configuration is as follows: &quot;column&quot;: [&quot;*&quot;]. You can configure the column field as follows:</td>
<td>Yes</td>
<td>Read all according to string type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fieldDelimiter</td>
<td>The delimiter used to separate the read fields.</td>
<td>Yes</td>
<td>,</td>
</tr>
<tr>
<td>Skipheader</td>
<td>The header of a file in CSV-like format is skipped if it is a title. Headers are not skipped by default. skipHeader is not supported for file compression.</td>
<td>No</td>
<td>False</td>
</tr>
<tr>
<td>encoding</td>
<td>Encoding of the read files.</td>
<td>No</td>
<td>utf-8</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default value</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| `nullFormat`  | Defining null (null pointer) with a standard string is not allowed in text files. Data synchronization provides nullFormat to define which strings can be expressed as null.  
For example, when `nullFormat:“null”`, is configured, if the source data is "null", it is considered as a null field in data synchronization. | No      | N/A           |
| `markDoneFileName` | The name of the file marked as "done". Check MarkDoneFile before data synchronization. If the file does not exist, wait for a while and check again. If the file exists, start the data synchronization task. | No      | N/A           |
| `MaxRetryTime` | The number of attempts made to check MarkDoneFile. The default value is 60. Try every minute for a duration of 60 minutes.                                                                                     | No      | 600           |
| `csvReaderConfig` | Reads the CSV files parameter configurations. It is the Map type. This reading is performed by the CsvReader for reading CSV files and involves many configuration items. Not configured items will use default settings. | No      | N/A           |
| `fileFormat`  | The read file type. By default, the file is read as a CVS file and the file content is parsed to a logical two-dimensional table for processing. If you set this file to binary, the file is copied and transmitted in the binary format. Such setting is applicable for peer-to-peer copy of directories between FTP and OSS files. Generally, you do not need to configure this item. | No      | N/A           |
Development in wizard mode

1. Choose source

Configure the data source and destination for the synchronization task.

Configurations:

- **Data source**: The datasource in the preceding parameter description. Enter the configured data source name.
- **File path**: The path in the above parameter description.
- **Column delimiter**: The fieldDelimiter in the preceding parameter description, which defaults to a comma (,).
- **Encoding format**: Encoding in the preceding parameter description, which defaults to utf-8.
- **null value**: nullFormat in the preceding parameter description to define a string that represents the null value.
- **Compression format**: Compress in the preceding parameter description, which defaults to "no compression".
- **Whether to include the table header**: skipHeader in the preceding parameter description, which defaults to "No".
2. **Field mapping**: The column in the preceding parameter description.

The source table field on the left and the target table field on the right are one-to-one correspondences, click Add row to add a single field and click Delete to delete the current field.

- **In-row mapping**: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.
- **Manually edit source table field**: Manually edit the fields, and each line indicates a field. The first and end blank lines are ignored.

3. **Channel control**

   **Configurations:**

   - **Concurrent job count**: Maximum number of threads used to concurrently read data from or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
   - **The maximum number of errors indicates the maximum number of dirty data records.**
   - **Task resource group**: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource, it is recommended that you add a Custom Resource Group. For more information, see *Add scheduling resources*. 

---

**Issue: 20191220**
Configure a synchronous Extraction Data job from the FTP database.

```json
{
  "type": "job",  // Indicates the version.
  "version": "2.0"

  "steps": [
    {
      "stepType": "ftp",  // plug-in name
      "parameter": {
        "path": [],  // File path
        "nullFormat": "",  // Null Value
        "compress": "",  // compression format
        "datasource": "",  // Data Source
        "column": []  // Field
      },
      "name": "Reader ",
      "category": "reader"
    },
    {
      "stepType": "stream ",
      "parameter": {}
    },
    "setting": {
      "errorLimit": {
        "record": "0"  // Number of error records
      },
      "speed": {
        "throttle": false,  // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
        "concurrent": "1",  // Number of concurrent tasks
        "dmu": 1  // DMU Value
      }
    },
    "order": {
      "hops": [
        {
          "from": "Reader ",
          "to": "Writer"
        }
      ]
    }
  ]
}
```
1.6.4.11 Configure Table Store (OTS) Reader

This topic describes data types and parameters supported by OTS Reader and how to configure Reader in script mode.

The OTS Reader plug-in provides the ability to read data from Table Store (OTS), which allows incremental data extraction within the specified data extraction range. Currently, the following three extraction methods are supported:

- Full table extraction
- Specified range extraction
- Specified partition extraction

Table Store is a NoSQL database service built on Alibaba Cloud's Apsara distributed system, enabling you to store and access massive structured data in real time. Table Store organizes data into instances and tables. Using data partition and Server Load Balancing (SLB) technology to provide seamless scaling.

In short, OTS Reader connects to OTS server by using the official Table Store Java SDK. It reads and transfers data to data synchronization field information, according to official data synchronization protocol standard, and then transmits the information to downstream Writer side.

Based on the Table Store table range, the OTS Reader divides the range into multiple tasks according to the number of data synchronization concurrencies. Each task is implemented with an OTS Reader thread.

Currently, OTS Reader supports all Table Store types. The Table Store conversion types in the OTS Reader is as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>MySQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Integer</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
</tr>
<tr>
<td>String type</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Note:
Table Store does not support "date" type. The long value is generally used as Unix TimeStamp at application layer when an error is reported.

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>The OTS server (service address) endpoint. For more information, see Endpoint.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>accessId</td>
<td>The accessId of the Table Store.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>accessKey</td>
<td>The accessKey of the Table Store.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Instance name</td>
<td>The Table Store instance name. The instance is an entity for using and managing OTS services. After you enable the Table Store service, you can create an instance in the console to create and manage tables. The instance is the basic unit for Table Store resource management. All access control and resource measurements performed by the Table Store for applications are completed at the instance level.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table.</td>
<td>The name of the extracted table. Only one table can be entered. The multi-table synchronization is not required for Table Store.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| column    | The column name set for synchronization in the configured table. The field information is described with JSON arrays because the Table Store is a NoSQL system. The corresponding field name must be specified when the OTS Reader extracts data.  
  - Reading of ordinary columns is supported, for example, `{"name":"col1"}`.
  - Reading of partial columns is supported. OTS Reader does not read unconfigured columns.
  - Reading of constant columns is supported, for example, `{"type":"STRING", "value" : "DataX"}`. The "type" is used to describe constant types. Currently, supported types include STRING, INT, DOUBLE, BOOL, BINARY (where the entered value is encoded with Base64), INF_MIN (the minimum system limit value for Table Store. You cannot enter the attribute value if this value is specified, otherwise an error may occur), and INF_MAX (maximum system limit value for Table Store. You cannot enter the value attribute if this value is specified, otherwise an error may occur).
  - Function or custom expression is not supported because the Table Store does not provide functions or expressions similar to SQL, and OTS Reader does not provide function or expression either. | Yes | N/A |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
</table>
| begin/end | This configuration item that must be used in pairs allows data to be extracted from the OTS table range. The "begin/end" describes the distribution of OTS PrimaryKeys within the range, which must cover all PrimaryKeys. The PrimaryKeys range under the OTS table requires to be specified. For the range with infinite limit, use ["type":"INF_MIN"] and ["type":"INF_MAX"]. For example, if you want to extract data from an OTS table with the primary keys [DeviceID, SellerID], begin/end is configured as follows:

"range":{
    "begin":[
      {"Type": "inf_min"}, // specify the minimum value of ergonomic ID
    ],
    "end":[
      {"type": "INF_MAX"}, // specify the maximum value for ergonomic ID
    ]
}

To extract data from the entire table, use the following configuration:

"range":{
    "begin":[
      {"type": "INF_MIN"}, // specify the minimum value of ergonomic ID
    ],
    "end":[
      {"type": "INF_MAX"}, // specify the maximum value for ergonomic deviceID
    ]
}

| split | This is an advanced configuration item for custom splitting, which we generally do not recommend. The custom splitting rule is generally applied when OTS Reader's auto splitting policy is invalid in the hotspot where the Table Store data is stored. "split" specifies a splitting point between Begin and End, and only the information of splitting point for partitionKey, which means that only the |
|-------|-------------------------------------------------|-----------| N/A   |
Development in script mode

Configure a job to extract data synchronously from the entire Table Store table to local machine.

```json
{
  "type": "job",
  "version": "2.0", //Indicates the version.
  "steps": [
    {
      "stepType": "ots", //plug-in name
      "parameter": {
        "datasource": "", //Data Source
        "column": [ // Field
          { "name": "column1" } // field name
        ],
        "range": {
          "split": [ // Split range
            { "type": "INF_MIN" },
            { "type": "STRING", "value": "splitPoint1" },
            { "type": "STRING", "value": "splitPoint2" },
            { "type": "STRING", "value": "splitPoint3" },
            { "type": "INF_MAX" }
          ],
          "end": [ // End range
            { "type": "INF_MAX" },
            { "type": "INF_MAX" },
            { "type": "STRING", "value": "end1" }
          ]
        }
      }
    }
  ]
}
```
{
    "type": "INT",
    "Value": "100"
},
"begin": [
    {
        "type": "INF_MIN"
    },
    {
        "type": "INF_MIN"
    },
    {
        "type": "STRING",
        "value": "begin1"
    },
    {
        "type": "INT",
        "value": "0"
    }
],
"table": "// table name",
"name": "Reader",
"category": "reader"
}

{//The following is a writer template. You can find the corresponding writer plug-in documentations.
    "stepType": "stream",
    "parameter": {},
    "name": "writer",
    "category": "writer"
},
"setting": {
    "errorLimit": {
        "record": "0" // Number of error records
    },
    "speed": {
        "throttle": false, // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
        "concurrent": "1" // Number of concurrent tasks
    },
    "dmu": 1 // DMU Value
},
"order": {
    "hops": [
        {
            "from": "Reader",
            "to": "Writer"
        }
    ]
}
1.6.4.12 Configuring PostgreSQL Reader

In this topic we will describe the data types and parameters supported by PostgreSQL Reader and how to configure the Reader in both wizard and script mode.

The PostgreSQL Reader plug-in reads data from PostgreSQL databases. At the underlying implementation level, the PostgreSQL Reader connects to a remote PostgreSQL database through JDBC and runs SELECT statements to extract data from the database. On the public cloud, RDS provides a PostgreSQL storage engine.

Specifically, PostgreSQL Reader connects to a remote PostgreSQL database through the JDBC connector. The SELECT SQL query statements are generated and sent to the remote PostgreSQL database based on your configuration. Then, the SQL statements are run and the returned results are assembled into abstract datasets using the custom data types of data integration. Datasets are passed to the downstream writer for processing.

- PostgreSQL Reader concatenates the table, column, and WHERE information you configured into SQL statements, and sends them to the PostgreSQL database.
- PostgreSQL directly sends the configured querySQL information to the PostgreSQL database.

Type conversion list

PostgreSQL Reader supports most data types in PostgreSQL. Check whether your data type is supported.

The PostgreSQL reader has a list of Type transformations for PostgreSQL, as shown below.

<table>
<thead>
<tr>
<th>Category</th>
<th>PostgreSQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>bigint, bigserial, integer, smallint, and serial</td>
</tr>
<tr>
<td>Floating point</td>
<td>double precision, money, numeric, and real</td>
</tr>
<tr>
<td>String</td>
<td>varchar, char, text, bit, and inet</td>
</tr>
<tr>
<td>Date and time</td>
<td>date, time, and timestamp</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Category</td>
<td>PostgreSQL data type</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Binary</td>
<td>bytea</td>
</tr>
</tbody>
</table>

**Note:**

- Except the preceding field types, other types are not supported.
- For "money", "inet", and "bit", you need to use syntaxes, such as "a_inet::varchar" to convert data types.

### Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The column name set to be synchronized in the configured table.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>Field information is described with JSON arrays. [*] indicates all columns by default.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• Column pruning is supported which means you can select some columns to export.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Change of column order is supported, which means you can export the columns in an order different from the table schema order.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Constant configuration is supported. You must follow the MySQL SQL syntax format, for example <code>[[&quot;id&quot;, &quot;table&quot;,&quot;1&quot;, &quot;'mingya.wmy'&quot;, &quot;'null'&quot;, &quot;to_char(a+1)&quot;, &quot;2.3&quot;, &quot;true&quot;]].</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- ID is normal column name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Table is a column name that contains Reserved Words</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 1 For plastic digital Constants</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 'mingya.wmy' is a String constant (note that a pair of single quotes is required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Null is a null pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Char_length (s) is the computed String Length Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 2.3 is a floating point number</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- True is a Boolean Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Column must contain the specified column set to be synchronized and it cannot be blank.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| SplitPk   | - If you specify the SplitPk when using PostgreSQLReader to extract data, it means that you want to use the fields represented by the SplitPk for data sharding. In this case, the Data Integration initiates concurrent jobs to synchronize data, which greatly improves the data synchronization efficiency.  
  · If you are using SplitPk, we recommend that you use the tables primary keys because the primary keys are generally even and data hot spots are less prone to split data fragments.  
  · Currently, SplitPk only supports data sharding for integer data types. Other types such as string, floating point, and date are not supported. If you specify an unsupported data type, the SplitPk is ignored and the data is synchronized using a single channel.  
  · If the SplitPk is not specified, the table data is synchronized using a single channel, for example, SplitPk is not provided or SplitPk value is null. | No       | N/A           |
| where     | PostgreSQLReader concatenates an SQL statement based on the specified column, table, and WHERE statement and extracts data, according to the SQL statement. For example, you can set the WHERE statement during a test. In actual service scenarios, the data on the current day are usually required to be synchronized, in which case you can set the WHERE statement as id > 2 and sex = 1.  
  · The WHERE statement can be effectively used for incremental synchronization.  
  · If the WHERE statement is not set or is left null, the full table data synchronization is applied. | No       | N/A           |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>querySQL (only available in advanced mode)</td>
<td>In business scenarios, where the WHERE statement is insufficient for filtering. In such cases, the user can customize a filter SQL using this configuration item. When this item is configured, the data synchronization system filters data using this configuration item directly instead of configuration items as tables, columns, and SplitPk. For example, for data synchronization after multi-table join, use <code>select a,b from table_a join table_b on table_a.id = table_b .id</code>. When querySQL is configured, PostgreSQL Reader directly ignores the configuration of table, column, and WHERE conditions.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Fetchsize</td>
<td>It defines batch data pieces that the plug-in and database server can fetch each time. The value determines the number of network interactions between the DataX system and the server, which can greatly improve data extraction performance. Note: The fetchsize value (&gt; 2048) may cause the data synchronization process Out of Memory (OOM).</td>
<td>No</td>
<td>512 MB</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

Configure the source and destination of the data for the synchronization task.

![Data Source Configuration](image)

Configurations:

- Data source: The datasource in the preceding parameter description. Enter the data source name you configured.
- Table: Table in the preceding parameter description. Select the table for synchronization.
- Data filtering: You are about to synchronize the filtering criteria for data, and limit keyword filtering is not supported for the time being. The SQL syntax is consistent with the selected data source.
- Shard key: You can use a column in the source data table as a shard key. It is recommended that you use a primary key or an indexed column as a shard key, and that only fields of type Integer are supported.

During data reading, the data split is based on configured fields to achieve concurrent reading, and improving data synchronization efficiency.

Note:
The shard key configuration is related to the source selection in data synchronization. The shard key configuration item is displayed only when you configure the data source.

2. The field mapping which is the column in the above parameter description.

The source table field on the left and the target table field on the right are one-to-one relationships, click Add row to add a single field and click Delete to delete the current field.

- Peer mapping: Click Enable Same-Line Mapping to establish a corresponding mapping relationship in the peer, note that match the data type.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.
- Manually edit source table field: Manually edit the fields where each line indicates a field. The first and end blank lines are ignored.

The function of adding a row is as follows:

- You can enter constants. The value must be enclosed by a pair of single quotes, such as 'abc' and '123'.
- Use this function with scheduling parameters, such as ${bizdate}.
- You can enter functions supported by relational databases, such as now() and count(1).
- If the value you entered cannot be parsed, the type is displayed as unidentified.
3. Control the tunnel

Configurations:

- DMU: A unit which measures the resources, including CPU, memory, and network bandwidth consumed during data integration. One DMU represents the minimum amount of resources used for a data synchronization task.

- Concurrent job count: The maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.

- The maximum number of errors indicates the maximum number of dirty data records.

- Task resource group: The machine on which the task runs, if the task number is large, the default Resource Group is used to wait for a resource. It is recommended that you add a Custom Resource Group. For more information, see Add scheduling resources.

Development in script mode

Configure a job to synchronously extract data from a PostgreSQL database.

```json
{
  "type": "job",
  "version": "2.0"} //Indicates the version
"steps": [
  {
    "stepType": "postgresql",/plug-in name
    "parameter": {
      "datasource": ", // Data Source
      "column": [// Field
        "col1",
        "col2"
      ],
      "where": ", //Filtering condition
      "splitPk": ",/using the fields represented by splitpk
      for Data Division, data Synchronization thus starts concurrent tasks
      for Data Synchronization
      "table": ", // table name
    },
```
"name": "Reader ",
"category": "reader"
},
{ //The following is a reader template. You can find corresponding writer plug-in documentations
"stepType": "stream ",
"parameter":{},
"name": "Writer ",
"category": "writer"
}
],
"setting":{
"errorLimit": {
"record": "0" //Number of error records
},
"speed": {
"throttle":false //False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
"concurrent": "1" //Number of concurrent tasks
"dmu": 1 //DMU Value
}
},
"order":{
"hops":[
{"from": "Reader ",
"to": "Writer"
}
]
}

Additional instructions

Active/standby synchronous data recovery problem

Active/standby synchronization means that PostgreSQL uses an active/standby disaster recovery mode in which the standby database continuously restores data from the active database through binlog. Because of time differences in the active/standby data synchronization, especially in some situations, such as network latency. The restored data in the standby database after synchronization is significantly different from the primary database data, that is to say, the data synchronized from the backup database is not a full image of the primary database of the current time.

If the data integration system synchronizes RDS data provided by Alibaba Cloud, the data can be directly read from the primary database without data restoration issues. However, this may cause issues on the master database load. Configure the data integration system properly for throttling.

Consistency constraints
PostgreSQL is an RDBMS data storage system, which can provide APIs for querying strong consistency data. For example, if another data writer writes data to the database during a synchronization task, PostgreSQL Reader does not obtain the newly written data because of the database snapshot features. For database snapshot characteristics, see MVCC Wikipedia.

The preceding paragraph lists all characteristics of data synchronization consistency under the PostgreSQL reader single-threaded model. Because PostgreSQL reader can use Concurrent Data Extraction based on your configuration information, therefore, data consistency cannot be strictly guaranteed. When the PostgreSQL reader is split based on the splitPk data, multiple concurrent tasks are initiated to complete the data synchronization. Since multiple concurrent tasks do not belong to the same read transaction, there are time intervals for multiple concurrent tasks at the same time, therefore, this data is an incomplete and inconsistent data snapshot.

Multi-threaded consistent snapshot requirements can only be solved from an engineering perspective. The following are engineering methods and solutions for different application scenarios.

- Single-threaded synchronization without data sharding. This method is slow but can ensure robust data consistency.
- Close other data writers to ensure the current data is static. For example, you can lock the table or close standby database synchronization. The disadvantage with this method is it may affect online businesses.

Database coding problem

PostgreSQL supports EUC_CN and UTF-8 encoding for simplified Chinese. PostgreSQL Reader extracts data using JDBC at the underlying level. JDBC is applicable for all types of encodings and can complete the transcoding at the underlying level. Therefore, PostgreSQL Reader can acquire the encoding and complete transcoding automatically without the need to specify the encoding.

PostgreSQL Reader cannot identify the inconsistency between the encoding written to the underlying layer of PostgreSQL and the configured encoding, nor provides a solution. Due to this issue, the exported codes may contain junk codes.

Incremental synchronization
PostgreSQL Reader uses a JDBC select statement for data extraction, so you can use select... WHERE... in either of the following ways:

- When online database applications write data into the database, the modify field is filled with the modification timestamp, including addition, update, and deletion (logical deletion). For this type of applications, PostgreSQL Reader only requires the WHERE statement followed by the timestamp of the last synchronization phase.
- For new streamline data, PostgreSQL Reader requires the WHERE statement followed by the maximum auto-increment ID of the last synchronization phase.

In the case that no fields are provided for the business to identify the addition or modification of data, PostgreSQL Reader cannot perform incremental data synchronization, and can only perform full data synchronization.

SQL Security

PostgreSQL Reader provides querySQL statements for you to SELECT data. PostgreSQL Reader does not perform security verification on querySQL.

1.6.4.13 Configuring SQL server Reader

This topic describes data types and parameters supported by the SQL server Reader and how to configure Reader in both wizard mode and script mode.

The SQL Server Reader plug-in provides the ability to read data from the SQL Server. At the underlying implementation level, the SQL Server Reader connects to a remote SQL Server database through JDBC and runs SELECT statements to extract data from the database.

Specifically, the SQL Server Reader connects to a remote SQL Server database through the JDBC connector. The SELECT SQL query statements are generated and sent to the remote SQL Server database based on your configuration. Then, the SQL statements are runned and the returned results are assembled into abstract datasets using the custom data types of data integration. Datasets are passed to the downstream writer for processing.

- SQL Server Reader concatenates the table, column, and the WHERE information you configured into SQL statements and sends them to the SQL Server database.
- SQL Server directly sends the querySQL information you configured to the SQL Server database.
SQL Server Reader supports most data types in SQL Server. Check whether your data type is supported.

SQL Server Reader converts SQL Server data types as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>SQL server data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>bigint, int, smallint, and tinyint</td>
</tr>
<tr>
<td>Float</td>
<td>float, decimal, real, and numeric</td>
</tr>
<tr>
<td>String type</td>
<td>char, nchar, ntext, nvarchar, text, varchar, nvarchar (MAX), and varchar (MAX)</td>
</tr>
<tr>
<td>Date and time type</td>
<td>date, datetime, and time</td>
</tr>
<tr>
<td>Boolean</td>
<td>bit</td>
</tr>
<tr>
<td>Binary, varbinary, varbinary (MAX), and timestamp</td>
<td>Binary, varbinary, varbinary (max), and timestamp</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table.</td>
<td>The table selected for synchronization. One job can only synchronize one table.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The column name set to be synchronized in the configured table. Field information is described with JSON arrays. [&quot;&quot;] indicates all columns by default.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- Column pruning is supported, which means you can select some columns to export.
- Change column order is supported, which means you can export the columns in an order different from the table schema order.
- Constant configuration is supported. You must follow the MySQL SQL syntax format, for example \["id", "table", "1", "'mingya.wmy'", "'null'", "to_char(a + 1)", "2.3", "true"] .
  - ID is normal column name
  - Table is a column name that contains Reserved Words
  - 1 For plastic digital Constants
  - 'mingya.wmy' is a String constant (note that a pair of single quotes is required)
  - null refers to the null pointer
  - to_char(a + 1) is a function expression
  - 2.3 is a floating point number
  - true is a Boolean value
- Column must contain the specified column set to be synchronized and it cannot be blank.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| splitPk   | If you specify the splitPk when using SQL Server Reader to extract data, it means the fields are represented by splitPk for data sharding. Then, the data synchronization system starts concurrent tasks to synchronize data, which greatly improves the data synchronization efficiency.  
  • We recommend that splitPk users use the tables primary keys because the primary keys are generally even and the data hot spots are less prone to split data fragments.  
  • Currently, splitPk only supports data sharding for integer data types. Other types such as float point, string, and date are not supported. If you specify an unsupported data type, SQL Server Reader reports an error. | No | N/A |
| where     | The filtering condition. The SQL Server Reader concatenates an SQL command based on the specified column, table, and WHERE statement and extracts data according to the SQL command. For example, you can specify the WHERE statement as limit 10 during a test. In actual business scenarios, the data on the current day is usually required to be synchronized. You can specify the WHERE statement as gmt_create > $bizdate.  
  • The WHERE statement can be effectively used for incremental synchronization.  
  • The WHERE statement can be effectively used for incremental synchronization. If the value is null, it means synchronizing all the information in the table. | No | N/A |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>querySQL</td>
<td>In some business scenarios, the WHERE statement is insufficient for filtering. In such cases, you can customize a filter SQL statement using this configuration item. When this item is configured, the data synchronization system filters data using this configuration item directly instead of configuration items, such as table and column. For example, for data synchronization after multi-table join, use <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. When querySQL is configured, SQL Server Reader directly ignores the configuration of table, column, and WHERE statements.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>fetchSize</td>
<td>It defines the pieces of batch data that the plug-in and database server can fetch each time. The value determines the number of network interactions between the data synchronization system and the server, which can greatly improve data extraction performance.</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>

Note:
A value greater than 2048 may lead to Out of Memory (OOM) for data synchronization.
Development in wizard mode

1. **Choose source**

   **Data source and destination**

   ![Wizard Mode Screenshot]

   **Configurations:**

   - **Data source:** The data source in the preceding parameter description. Enter the data source name you configured.
   - **Table:** The table in the preceding parameter description. Select the table for synchronization.
   - **Filtering condition:** You should synchronize the data filtering conditions. Limit keyword filter is not supported yet. SQL syntaxes vary with data sources.
   - **Shard key:** You can use a column in the source table as the shard key. It is recommended to use a primary key or an indexed column as the shard key.
2. Field mapping: The column in the preceding parameter description.

The source table field on the left and the target table field on the right are one-to-one relationships, click Add row to add a single field and click Delete to delete the current field.

- Peer mapping: Click Enable Same-Line Mapping to establish a corresponding mapping relationship in the peer that matches the data type.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.
- Manually edit source table field: Manually edit the fields where each line indicates a field. The first and end blank lines are ignored.

The function of adding a row is as follows:

- You can enter constants. Each constant must be enclosed in a pair of single quotes, such as 'abc' and '123'.
- Use this function with scheduling parameters, such as ${bizdate}$.
- Enter functions supported by relational databases, such as now() and count(1).
- If the value you entered cannot be parsed, the type is displayed as 'Not Identified'.

3. Channel control

Configurations:

- **DMU**: A unit which measures the resources consumed during data integration, including CPU, memory, and network bandwidth. One DMU represents the minimum amount of resources used for a data synchronization task.

- **Concurrent count**: The maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.

- **The maximum number of errors indicates the maximum number of dirty data records.**

- **Task resource group**: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend that you add a Custom Resource Group. For more information, see [Add scheduling resources](#).

Development in script mode

**Configure a job to synchronously extract data from an SQL Server database:**

```json
{
    "type": "job",
    "version": "2.0" // Indicates the version.
    "steps": [
        {
            "stepType": "SQL Server", // Plugin name
            "parameter": {
                "datasource": "", // Data Source
                "column": [ // Column name
                    "id",
                    "name"
                ],
                "where": "", // Filtering condition
                "splitPk": "", // If split PK is specified, indicates that you want to slice the data using the fields represented by splitpk
                "table": "" // Data Sheet
            }
        }
    ]
}
```
"name": "Reader ",
"category": "Reader"
},
{"//The following is a writer template. You can find the corresponding writer plug-in documentations.
"stepType": "stream ",
"parameter":{}
"name": "writer",
"category": "writer"
}
],
"setting":{
"errorLimit": {
"record": "0"//Number of error records
},
"speed": {
"throttle":false,//False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
"concurrent": "1",//Number of concurrent tasks
"dmu": 1 // DMU Value
}
},
"order":{
"hops":[
{
"from": "Reader ",
"to": "Writer"
}
]
}
}

Additional instructions

**Active/standby synchronous data recovery problem**

Active/standby synchronization means the SQL Server uses a active/standby disaster recovery mode in which the standby database continuously restores data from the master database through binlog. Due to the time difference in the primary /backup data synchronization, especially in situations such as network latency, the restored data in the backup database after synchronization is significantly different from the primary database data, that is to say, the data synchronized from the backup database is not a full image of the primary database at the current time.

If the data integration system synchronizes RDS data provided by Alibaba Cloud, the data is directly read from the primary database without data restoration concerns. However, this may cause concerns on the master database load and configure it properly for throttling.

**Consistency constraints**
SQL Server is an RDBMS system in terms of data storage, which can provide APIs for querying strong consistency data. For example, if another data writer writes data to the database during a synchronization task, SQL Server Reader does not obtain the newly written data because of the database snapshot features. For more information on the database snapshot features, refer to the [MVCC Wikipedia](https://en.wikipedia.org/wiki/Multiversion_concurrency_control).

The preceding paragraph lists the characteristics of data synchronization consistency under the SQL Server reader single-threaded model. Data consistency cannot be guaranteed because the SQL Server reader can use Concurrent Data Extraction based on your configuration information. When the SQL Server reader is split based on the splitPk data, multiple concurrent tasks are initiated to complete the synchronization of data. Since multiple concurrent tasks do not belong to the same read transaction, and time intervals for multiple concurrent tasks exist at the same time, therefore, this data is an incomplete and inconsistent snapshot of the data.

Multi-threaded consistent snapshot can only be solved from an engineering perspective. The following are engineering method and solutions for different application scenarios.

- Use single-threaded synchronization without data sharding. This is slow but can ensure robust data consistency.
- Close other data writers to ensure the current data is static. For example, you can lock the table or close the standby database synchronization. However, the disadvantage is online businesses may be affected.

**Database coding problem**

The SQL Server Reader extracts data using JDBC at the underlying level. JDBC is applicable to all types of encodings and can complete transcoding at the underlying level. Therefore, SQL Server Reader can identify the encoding and complete transcoding automatically without the need to specify the encoding.

**Incremental synchronization**

SQL Server reader uses a JDBC SELECT statement for data extraction, so you can use select... Where... in either of the following ways:

- When online database applications write data into the database, the modify field is filled with the modification timestamp, including addition, update, and deletion (logical deletion). For this type of applications, SQL Server Reader only
requires the WHERE statement followed by the timestamp of the last synchronization phase.

- For new streamline data, SQL Server Reader requires the WHERE statement followed by the maximum auto-increment ID of the last synchronization phase.

In case no field is provided for the business to identify the addition or modification of data, SQL Server Reader cannot perform incremental data synchronization and can only perform full data synchronization.

SQL security

SQL Server Reader provides querySQL statements for you to SELECT data. The SQL Server Reader conducts no security verification on querySQL. The security during use is ensured by the data synchronization users.

1.6.4.14 Configure LogHub Reader

In this topic we will describe the data types and parameters supported by LogHub Reader and how to configure Reader in both wizard and script mode.

Honed originally by the Big Data demands of Alibaba Group, Log Service (or "LOG" for short, formerly “SLS”) is an all-in-one service for real-time data. With its capabilities to collect, consume, deliver, query, and analyze log-type data, Log Service allows you to process and analyze massive amounts of data much more efficiently. LogHub Reader uses the Java SDK of the Log Service to consume real-time log data in LogHub, and convert the log data to the Data Integration transfer protocol and sends the converted data to Writer.

Implementation

LogHub Reader consumes real-time log data in LogHub by using the following version of Log Service Java SDK:

```xml
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>aliyun-log</artifactId>
  <version>0.6.7</version>
</dependency>
```

Logstore is a component of the Log Service for collecting, storing, and querying log data. Logstore read and write logs are stored on a shard. Each log library consists of several partitions, each consists the left closed right open interval of MD5, each interval range is not covered by each other, and the range of all the intervals is the
entire MD5 range of values, each partition can provide a certain level of service capability.

- Writing: 5 MB/s, 2000 times/s.
- Read: 10 MB/s, 100 times/s.

LogHub Reader consumes logs in shards, and the detailed consumption process (GetCursor and BatchGetLog-related APIs) is as follows:

- Obtains a cursor based on the interval range.
- Reads logs based on the cursor and step parameters and returns the next cursor.
- Moves the cursor continuously to consume logs.
- Splits tasks by shard for concurrent execution.

LogHub Reader supports LogHub type conversion, as shown in the following table:

<table>
<thead>
<tr>
<th>Datax internal type</th>
<th>Loghub data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>String</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>The Log Service endpoint is a URL for accessing a project and its internal log data. It is associated with the Alibaba Cloud region and name of the project. Service entry for each region, see service entry.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>accessId</td>
<td>It refers to an AccessKey for accessing the Log Service, which is used to identify the accessing user.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>accessKey</td>
<td>It refers to another AccessKey for accessing the Log Service, which is used to verify the user's key.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>project</td>
<td>It refers to the project name of the target Log Service, which is the resource management component in the Log Service for isolating and controlling resources.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>logstore</td>
<td>It refers to the name of the target Logstore. Logstore is a component of the Log Service for collecting, storing, and querying log data.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>batchSize</td>
<td>It refers to the number of data entries queried from the Log Service at a time.</td>
<td>No</td>
<td>128</td>
</tr>
<tr>
<td>column</td>
<td>Column names in each data entry. Here, you can set a metadata item in the Log Service as the synchronization column. Supported metadata items include &quot;Topic&quot;, &quot;MachineUUID&quot;, &quot;HostName&quot;, &quot;Path&quot;, and &quot;LogTime&quot;, which represents the log topic, unique identifier of the collection machine, host name, path, and log time, respectively. The sub-table represents the log theme, the acquisition machine uniquely identified, the host name, path, log time, and so on.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>BeginDateTime</td>
<td>Start time of data consumption. The parameter defines the left border of a time range (left closed and right open) in the format of yyyyMMddHHmmss (such as 20180111013000), and can work with the scheduling time parameter in DataWorks.</td>
<td>Required</td>
<td>Blank</td>
</tr>
</tbody>
</table>

**Note:** The values of fields in the format are case insensitive.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>enddatetime</td>
<td>The end time of the data consumption. The parameter defines the right border of a time range (left closed and right open) in the format of yyyyMMddHHmmss (such as 20180111013010) and can work with the scheduling time parameter in DataWorks.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The combination of enddatetime and maid is used together.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>begintimestampmillis</td>
<td>It refers to the start time of data consumption in milliseconds and is the left boundary of the time range (left-closed and right-open).</td>
<td>Required</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Begintimestampmillis and endtimestampmillis combination for use.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 represents the beginning of the log service cursor cursormode. Begin. The beginDateTime mode is recommended.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>endtimestampmillis</td>
<td>It refers to the end time of data consumption in milliseconds and is the right boundary of the time range (left-closed and right-open).</td>
<td>Required</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Endtimestampmillis and begintimestampmillis combination for use.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1 represents the last location of the log service cursor, cursormode.End. The endDateTime mode is recommended.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

Configure the source and destination of the data for the synchronization task.

Configurations:

- Data source: The data source in the preceding parameter description. Enter the data source name you configured.
- Log start time: The start time of data consumption. It defines the left border of a time range (left closed and right open) in the format of yyyymMddHHmmss, such as 20180111013000 and can work with the scheduling time parameter in DataWorks.
- Log end time: The end time of data consumption. It defines the right border of a time range (left closed and right open) in the format of yyyymMddHHmmss, such as 20180111013010 and can work with the scheduling time parameter in DataWorks.
2. The field mapping which is the column in the above parameter description.

The source table field on the left and the target table field on the right are one-to-one relationships, click Add row to add a single field and click Delete to delete the current field.

- **Peer mapping**: Click Enable Same-Line Mapping to establish a corresponding mapping relationship in the peer, note that match the data type.
- **Automatic formatting**: The fields are automatically sorted based on corresponding rules.
- **Manually edit source table field**: Manually edit the fields where each line indicates a field. The first and end blank lines are ignored.

The function of adding a row is as follows:

- **You can enter constants.** The value must be enclosed by a pair of single quotes, such as 'abc' and '123'.
- **Use this function with scheduling parameters**, such as `${bizdate}`.
- **You can enter functions supported by relational databases**, such as `now()` and `count(1)`.
- **If the value you entered cannot be parsed**, the type is displayed as Not identified.
3. Control the tunnel

Configurations:

- DMU: A unit which measures the resources consumed during data integration, including CPU, memory, and network bandwidth. One DMU represents the minimum amount of resources used for a data synchronization task.

- Concurrent job count: The maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.

- The maximum number of errors indicates the maximum number of dirty data records.

- Task resource group: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend you add a Custom Resource Group. For more information, see #unique_12.

Development in script mode

The following is a script configuration sample. For details about parameters, see the preceding Parameter Description.

```json
{
  "type": "job",
  "version": "1.0"} //Indicates the version.
  "steps": [
    {
      "stepType": "loghub", // plug-in name
      "parameter": {
        "datasource": "", // Data Source
        "column": "["col0", "col1", "col2", "col3", "col4", "topic"], // Field
        "hostname": "", // log theme
        "hostname": "", // host name
```
1.6.4.15 Configure OTSReader-Internal

This topic describes the data types and parameters supported by OTSReader-Internal and how to configure Reader in script mode.

Table Store (originally known as OTS) is a NoSQL database service built upon Alibaba Cloud's Apsara distributed system, enabling you to store and access massive structured data in real time. Table Store organizes data into instances and tables. Using data partition and Server Load Balancing (SLB) technology, it provides seamless scaling.

OTSReader-Internal is used to export table data for Table Store Internal model, while OTS Reader is used to export data for OTS Public model.
Table Store Internal model supports multi-version columns, so OTSReader-Internal also provides two data export modes:

- **Multi-version mode**: A version mode that exports data in multiple versions. Table Store supports multiple versions.

  Export solution: The Reader plug-in expands a cell of Table Store into a one-dimensional table consisting of four tuples: PrimaryKey (column 1-4), ColumnName, Timestamp, and Value (the principle is similar to the multi-version mode of HBase Reader). The four tuples are passed in to the Writer as four columns in Datax record.

- **Normal mode**: Consistent with the normal mode of the hbase reader, export the latest version of each column in each row of data. For more information, see [Configure HBase reader](#) the normal mode content that is supported by the hbase reader in.

In short, OTS Reader connects to Table Store's server and reads data through Table Store official Java SDK. OTS Reader optimizes the read process using features, such as read timeout retry and exceptional read retry.

Currently, OTS Reader supports all Table Store types. The conversion of Table Store types in the OTSReader-Internal is as follows:

<table>
<thead>
<tr>
<th>Data integration internal types</th>
<th>Table Store data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>Integer</td>
</tr>
<tr>
<td>Double</td>
<td>Double</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Bytes</td>
<td>Binary</td>
</tr>
</tbody>
</table>

**Parameter description**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>The plug-in operation mode, supporting normal and multiVersion, which refers to normal mode and multi-version mode respectively.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>endpoint</td>
<td>The EndPoint of Table Store Server.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>accessId</td>
<td>The access ID for Table Store.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>accessKey</td>
<td>The access key for Table Store.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Instance name</td>
<td>The Table Store instance name. The instance is an entity for using and managing Table Store service. After you enable the Table Store service, you can create an instance in the Console to create and manage tables. The instance is the basic unit for Table Store resource management. All access control and resource measurement performed by the Table Store for applications are completed at the instance level.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The name of the table to be extracted. Only one table can be filled in. Multi-table synchronization is not required for Table Store.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| Range        | The export range: [begin,end).  
  · When begin is less than end, reads data in positive sequence.  
  · When begin is greater than end, reads data in inverted sequence.  
  · Begin and end cannot be equal.  
  · The following types are supported: string, int, and binary. The binary data is passed in as Base64 strings in binary format. INF_MIN represents an infinitely small value and INF_MAX represents an infinitely large value. | No       | Reads from the beginning of the table to the end of the table |
### Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>range: &quot;begin &quot;]</td>
<td>The starting range that is exported. The value can be an empty array, a PK prefix, or a complete PK. When reading the data in positive order, the default fill PK suffix is inf_min, and the reverse order is inf_max, as shown in the example as follows.</td>
<td>No</td>
<td>Read data from the beginning of the table</td>
</tr>
</tbody>
</table>

If your table has two PrimaryKeys in the type of string and int, the table data can be entered using the following three methods:

- [] Indicates that it is read from the beginning of the table.
- ["type": "string", "value": "a"] means from ["type": "string", "value": "a"], ["type": "INF_MIN "].
- ["type": "string", "value": "a"], ["type": "INF_MIN"]

PrimaryKey column in binary type is special. JSON does not support directly passing in binary data, so the following rules are defined:

To pass in binary data, you must use (Java) Base64.encodeBase64String method to convert binary data into a visualized string and then enter the string in value. The example is as follows (Java):

- byte[] bytes = "hello".getBytes(); :Create binary data. Here the byte value of string hello is used.
- String inputValue = Base64.encodeBase64String(byte): Calls Base64 method to convert binary data into visualized strings.

Run the preceding code, and then the inputValue of "aGVsbG8=" can be obtained.

Finally, write the value into the configuration: ["type": "binary", "value": "aGVsbG8="].
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| range: {"end "} | The end range that is exported. The value can be an empty array, a PK prefix, or a complete PK. When reading data in positive order, the default population PK suffix is INF_MAX, and the reverse order is INF_MIN.  
If your table has two PKs in the type of string and int, the table data can be entered using the following three methods:  
  · [] Indicates that it is read from the beginning of the table.  
  · ["type":"string","value":"a"] means from ["type":"string","value":"a"],[ "type":"INF_MIN" ]].  
  · ["type":"string","value":"a"],[ "type":"INF_MIN" ]].  
   PrimaryKey column in binary type is special. JSON does not support directly passing in binary data, so the following rules are defined:  
To pass in binary data, you must use (Java) Base64.encodeBase64String method to convert binary data into a visualized string and then enter the string in value. The example is as follows (Java):  
  · byte[] bytes = "hello".getBytes(); Create binary data. Here the byte value of string hello is used.  
  · String inputValue = Base64.encodeBase64String(bytes): Call Base64 method to convert binary data into visualized strings.  
Run the preceding code, and then the inputValue of "aGVsbG8=" can be obtained.  
Finally, write the value into the configuration: {"type":"binary","value" : "aGVsbG8="}. | No        | Read to end of table |

Issue: 20191220
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| range: 
  split "}"  | If too much data needs to be exported, you can enable concurrent export. Split can split the data in the current range into multiple concurrent tasks according to split points. Note:  
  • The value entered in the split must be in the first column of PrimaryKey (partition key) and the value type must be consistent with that of the PartitionKey.  
  • The values range must be between begin and end.  
  • The value within the split must increase or decrease progressively depending on the positive and inverted relationship between begin and end. | No       | Empty cut point     |
| column          | Specifies the columns to export, supporting common and constant columns. Format (multi-version mode is supported)  
  Regular column format: 
  
  "name":"{your column name}"  
  |                                  |          |                    |
| timeRange       | The time range of the request data. The read range is [begin,end).  
  Note: Begin must be smaller than end. | No       | Read all versions by default |
<p>| timeRange       | The start time of the time range of request data. The value range is 0-LONG_MAX. | No       | 10 by default        |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeRange:{&quot;end&quot;} (only multi-version mode is supported)</td>
<td>The end time of the time range of request data. The value range is 0-LONG_MAX.</td>
<td>No</td>
<td>- Default value : Long Max(9223372036854775806L)</td>
</tr>
<tr>
<td>maxVersion (only multi-version mode is supported)</td>
<td>The request specified version. The value range is 1-INT32_MAX.</td>
<td>No</td>
<td>Reads all versions by default</td>
</tr>
</tbody>
</table>

Development in wizard mode

**Currently, development in wizard mode is not supported.**

Development in script mode

**Multi-version mode**

```json
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "otsreader-internalreader ",
            "parameter": {
                "mode": "multiversion ",
                "endpoint": "",
                "accessId": "",
                "accessKey": "",
                "instanceName": "",
                "table": "",
                "range": {
                    "begin": [
                        { "type": "string", "value": "a" },
                        { "type": "INF_MIN" }
                    ],
                    "end": [
                        { "type": "string", "value": "g" },
                        { "type": "INF_MAX" }
                    ]
                }
            }
        }
    }
}
```
"type": "INF_MAX"
},
"split": [
{
"type": "string",
"value": "b"
},
{
"type": "string",
"value": "c"
}
],
"column": [
{
"name": "attr1"
}
],
"timeRange": {
"begin": 1400000000,
"end": 1600000000
},
"maxVersion": 10
}
,"writer": {

Normal mode

{  "type": "job",
   "version": "1.0",
   "configuration": {
      "reader": {
         "plugin": "otsreader-internalreader ",
         "parameter": {
            "mode": "normal",
            "endpoint": "",
            "accessId": "",
            "accessKey": "",
            "instanceName": "",
            "table": "",
            "range": {
               "begin": [
                  {
                     "type": "string",
                     "value": "a"
                  },
                  {
                     "type": "INF_MIN"
                  }
               ],
               "end": [
                  {
                     "type": "string",
                     "value": "g"
                  },
                  {
                     "type": "INF_MAX"
                  }
               ]
         }
      }
   }
}
1.6.4.16 Configure OTSStream Reader
This topic describes the data types and parameters supported by OTSStream Reader and how to configure Reader in script mode.

OTSStream Reader plug-in is mainly used for exporting Table Store incremental data. Incremental data can be seen as operation logs which include data and operation information.

Different from full export plug-in, incremental export plug-in only has multi-version mode and it does not support specified columns. This is related to the
principle of incremental export. See the following for more information about export format.

Before using the plug-in, ensure that the Stream feature is enabled. You can enable the feature when creating the table or enable it using SDK UpdateTable API.

How to enable Stream:

```java
Syncclient client = new SyncClient("","","","");
Enable Stream when you create the table:
CreateTableRequest createTableRequest = new CreateTableRequest(tableMeta);
createTableRequest.setStreamSpecification(new StreamSpecification(true, 24)); // 24 means that the incremental data is retained for 24 hours
client.createTable(createTableRequest);
If Stream is not enabled when the table is created, you can enable it with UpdateTable:
UpdateTableRequest updateTableRequest = new UpdateTableRequest("tableName");
createTableRequest.setStreamSpecification(new StreamSpecification(true, 24)); // 24 means that the incremental data is retained for 24 hours
client.updateTable(updateTableRequest);
```

Implementation

You can enable Stream and set expiration time by using SDK UpdateTable feature to enable incremental feature. When incremental feature is enabled, Table Store server saves your operation logs additionally. Each partition has a sequential operation log queue. Each operation log is moved by garbage collection after a period of time which is the expiration time you specified.

Table Store SDK provides several stream-related APIs for reading these operation logs. The incremental plug-in also obtains incremental data with Table Store SDK API, transforms incremental data into multiple 6-tuples (pk, colName, version, colValue, opType, sequenceInfo), and imports them into MaxCompute.

The format of the export data

In Table Store multi-version mode, the table data format is in three-level mode, namely row > column > version. One row can have multiple columns. The column name is not fixed, and each column can have multiple versions. Each version has a specific timestamp (version number).

You can perform read/write operations with Table Store API. Table Store records incremental data by recording your recent write operations to the table (or data change operation). Therefore, incremental data can also be seen as a series of operation records.
Table Store has three types of data change operations: PutRow, UpdateRow, and DeleteRow:

- **PutRow**: Write a row. If the row already exists, it is overwritten.
- **UpdateRow**: Updates a row without changing other data of the original row. Update may include adding or overwriting (if the corresponding version of the corresponding column already exists) some column values, deleting all the versions of a column, and deleting a version of a column.
- **DeleteRow**: Delete a row.

Table Store generates corresponding incremental data records according to each type of operation. Reader plug-in reads the records and exports data in the format of Datax.

Because Table Store has the feature of dynamic column and multi-version, a row exported by Reader plug-in does not correspond to a row in Table Store, but a version of a column in Table Store. A row in Table Store can be exported as multiple rows. Each row includes primary key value, the column name, the timestamp of the version under the column (version number), the version value, and operation type. If isExportSequenceInfo is set as true, the time sequence information is also included.

When the data is transformed into Datax format, we define four types of operations as follows:

- **U (UPDATE)**: Writes a version of a column.
- **DO (DELETE_ONE_VERSION)**: Deletes a version of a column.
- **DA (DELETE_ALL_VERSION)**: Deletes all the versions of a column. Delete all versions of the corresponding column according to the primary key and column name.
- **DR (DELETE_ROW)**: Deletes a row. Deletes all data of the row according to primary key.

Assuming that the table has two primary key columns. The names of the two primary key columns are pkName1 and pkName2. The example is as follows:
<table>
<thead>
<tr>
<th>pkName1</th>
<th>pkName2</th>
<th>columnName</th>
<th>timestamp</th>
<th>columnValue</th>
<th>opType</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk1_V1</td>
<td>pk2_V1</td>
<td>col_a</td>
<td>1441803688001</td>
<td>col_val1</td>
<td>U</td>
</tr>
<tr>
<td>pk1_V1</td>
<td>pk2_V1</td>
<td>col_a</td>
<td>1441803688002</td>
<td>col_val2</td>
<td>U</td>
</tr>
<tr>
<td>pk1_V1</td>
<td>pk2_V1</td>
<td>col_b</td>
<td>1441803688003</td>
<td>col_val3</td>
<td>U</td>
</tr>
<tr>
<td>pk1_V2</td>
<td>pk2_V2</td>
<td>col_a</td>
<td>1441803688000</td>
<td>—</td>
<td>Do</td>
</tr>
<tr>
<td>pk1_V2</td>
<td>pk2_V2</td>
<td>col_b</td>
<td>—</td>
<td>—</td>
<td>Da</td>
</tr>
<tr>
<td>pk1_V3</td>
<td>pk2_V3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Dr</td>
</tr>
<tr>
<td>pk1_V3</td>
<td>pk2_V3</td>
<td>col_a</td>
<td>1441803688005</td>
<td>col_val1</td>
<td>U</td>
</tr>
</tbody>
</table>

Assuming that the export data has seven rows as shown in the preceding example, corresponding to the three rows in Table Store table. The primary keys are (pk1_V1, pk2_V1), (pk1_V2, pk2_V2), and (pk1_V3, pk2_V3).

- For the row whose primary key is (pk1_V1, pk2_V1), three operations are required, respectively writing two versions of col_a column and one version of col_b column.
- For the row whose primary key is (pk1_V2, pk2_V2), two operations are required, respectively deleting one version of col_a column and all versions of col_b column.
- For the row whose primary key is (pk1_V3, pk2_V3), two operations are required, respectively deleting the whole row and writing one version of col_a column.

Currently OTSStream Reader supports all OTS types. The conversion list for Table Store types is as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>OTSStream data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Integer</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
</tr>
<tr>
<td>String type</td>
<td>String-</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Type classification</td>
<td>OTSstream data type</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataSource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>dataTable</td>
<td>The table name from which the incremental data is exported. The table needs to enable the Stream feature. You can enable the feature when creating the table or enable it using UpdateTable API.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default value</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>statusTable</td>
<td>The name of the table used by the Reader plug-in to record the status, these States can be used to reduce scanning of data in non-target ranges to speed up export. statusTable is the table for recording status in Reader. If the table does not exist, Reader creates the table automatically. When an offline export task is completed, you must not delete the table. The statuses recorded in the table can be used for the next export task.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You only have to name the table, and do not have to create the table. Reader plug-in tries to create the table under your instance. If the table does not exist, it is created. If the table already exists, it judges whether the Meta of the table is consistent with expectation. If it is inconsistent, an exception is thrown.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When an export is completed, you must not delete the table. The statuses of the table can be used for the next export task.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• The table enables TTL and data expire automatically, therefore, we can consider that the data volume is small.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For the Reader configurations of different dataTables under one instance, you can use the same statusTable. The status messages recorded are independent of each other.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In conclusion, you must configure a name such as TableStoreStreamReaderStatusTable. Note that the name must not be a duplicate with that of business-related tables.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| startTimestampMillis | The left boundary of the time range of the incremental data (left closed right) in milliseconds.  
  - Reader finds the point corresponding to startTimestampMillis in statusTable, and reads and exports data from that point.  
  - If the corresponding point is not found in statusTable, the system reads from the first entry of the incremental data retained in the system and skips the data whose write time is earlier than startTimestampMillis. | No      | N/A           |
| endTimestampMillis    | The right border of the time range (left closed and right open) of incremental data in milliseconds.  
  - After exporting data from the point of startTimestampMillis, Reader finishes data export at the first entry of data whose timestamp is later than endTimestampMillis.  
  - When all the incremental data are read, the read is completed, even if endTimestampMillis is not reached. | No      | N/A           |
| date              | The data format is yyyyMMdd, for example 20151111. If you do not specify a date, you must specify a startTimestampMillis and endTimestampMillis, and also reversed. For example, Alibaba Cloud Data Process Center scheduling only supports day level. Therefore, the configuration function is similar to startTimestampMillis and endTimestampMillis. | No      | N/A           |
| isExportSequenceInfo | Determines whether to export the time sequence information. Time sequence information includes the data write time. The default value is false, which means not to export data. | No      | N/A           |
| maxRetries        | The maximum number of retries of each request when incremental data is read from TableStore. The default value is 30. There are intervals between retries. The total time of 30 retries is approximately 5 minutes, which generally does not require changes. | No      | N/A           |
DataWorks

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTimeStr</td>
<td>The left border of the time range (left closed and right open) of incremental data, in milliseconds (in the format of yyyymmddhh24miss).</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>endTimeStr</td>
<td>The right border of the time range (left closed and right open) of incremental data, in milliseconds (in the format of yyyymmddhh24miss).</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Development in wizard mode**

Currently, development in wizard mode is not supported.

**Development in script mode**

The following is a script configuration sample. For details about parameters, see the preceding Parameter Description.

```json
{
    "type": "job",
    "version": "2.0" //Indicates the version.
    "steps": [
        {
            "stepType": "otdsstream", // plug-in name
            "parameter": {
                "statusTable": "TableStoreStreamReaderStatusTable", // The name of the table for recording the status.
                "maxRetries": 30, // when you read incremental data from the tablestore, maximum number of retries per request, by default 30
                "isExportSequenceInfo": false, // do you want to export timing information?
                "datasource": "$ srcdatasource", // Data Source
                "startTimeString": "$ { starttime }", // The left boundary of the time range of the incremental data (left closed right open)
                "table": "ok", //Target table name
                "endTimeString": "$ {endtime}" // time range of incremental data (left closed right) right Border
            },
            "name": "Reader ",
            "category": "Reader"
        },
        {
            "stepType": "stream ",
            "parameter":{}
            "name": "Writer ",
            "category": "Writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": "0" //Number of error records
        },
```
1.6.4.17 Configure RDBMS Reader

This topic describes the data types and parameters supported by RDBMS Reader and how to configure Reader in script mode.

The RDBMS Reader plug-in allows you to read data from RDBMS (distributed RDS). At the underlying implementation level, RDBMS Reader connects to a remote RDBMS database through JDBC and runs corresponding SQL statements to SELECT data from the RDBMS database. Currently, it supports reading data from databases including DM, DB2, PPAS, and Sybase. Currently, the RDBMS plug-in is only adapted to the MySQL engine. RDBMS is a distributed MySQL database, and most of the communication protocols are applicable to MySQL use cases.

Specifically, RDBMS Reader connects to a remote RDBMS database through the JDBC connector. The SELECT SQL query statements are generated and sent to the remote RDBMS database based on your configuration. Then, the SQL statements are run and the returned results are assembled into abstract datasets using the custom data types of data synchronization. Datasets are passed to the downstream writer for processing.

RDBMS Reader concatenates the table, column, and WHERE information you configured into SQL statements and sends them to the RDBMS database. For the querySQL information that you configure, the RDBMS sends it directly to the RDBMS database.

RDBMS Reader supports the most generic rational database types, such as numbers and characters. Check whether your data type is supported and select a reader based on a specific database.
### Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| jdbcUrl   | Information of the JDBC connection to the opposite-end database. The format of jdbcUrl is in accordance with the RDBMS official specification, and the URL attachment control information can be entered. Note that JDBC formats vary with databases and DataX selects an appropriate database driver for data reading based on a specific JDBC format.  
  - DB2 jdbc:db2://ip:port/database  
  - PPAS jdbc:edb://ip:port/database | Yes | N/A |

RDBMS Writer adds new database support in the following ways.

- Enter the corresponding directory of RDBMSWriter. `${DATAX_HOME}` is the main directory of DataX, that is, `${DATAX_HOME}/plugin/writer/rdbmswriter`.
- Under the RDBMS Reader directory, you can find the plugin.json configuration file. Use this file to register your specific database driver, which is placed in the drivers array. The RDBMS Reader plug-in dynamically selects the appropriate database driver to connect to the database when executing the job.

```json
{
  "name": "RDBMS Reader",
  "class": "com.alibaba.datax.plugin.reader.RDBMS Reader.RDBMS Reader",
  "description": "useScene: prod. mechanism : Jdbc connection using the database, execute select sql, retrieve data from the ResultSet . warn: The more you know about the database , the less problems you encounter.",
  "developer": "alibaba",
  "drivers": [
    "dm.jdbc.driver.DmDriver",
    "com.ibm.db2.jcc.DB2Driver",
    "com.sybase.jdbc3.jdbc.SybDriver",
    "com.edb.Driver"
  ]
}
```

The RDBMS Reader directory contains the `libs` sub-directory, under which you need to put your specific database driver.

```
$tree
```

```bash
|-- libs
|  |-- Dm7JdbcDriver16.jar
|  |-- commons-collections-3.0.jar
|  |-- commons-io-2.4.jar
|  |-- commons-lang3-3.3.2.jar
|  |-- commons-math3-3.1.1.jar
|  |-- datax-common-0.0.1-SNAPSHOT.jar
|  |-- datax-service-face-1.0.23-20160120.024328-1.jar
|  |-- db2jcc4.jar
|  |-- druid-1.0.15.jar
|  |-- edb-jdbc16.jar
|  |-- fastjson-1.1.46.sec01.jar
|  |-- guava-r05.jar
|  |-- hamcrest-core-1.3.jar
|  |-- jconn3-1.0.0-SNAPSHOT.jar
|  |-- logback-classic-1.0.13.jar
|  |-- logback-core-1.0.13.jar
|  |-- plugin-rdbms-util-0.0.1-SNAPSHOT.jar
|  `-- slf4j-api-1.7.10.jar
`-- plugin.json
`-- plugin_job_template.json
`-- RDBMS Reader-0.0.1-SNAPSHOT.jar
```
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>The password corresponding to the specified username for the data source.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The selected table that needs to be synchronized.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| column    | The configured table requires a collection of column names that are synchronized, using a JSON array to describe the field information, all column configurations, such as ["*"], are used by default.  
  - Column pruning is supported, which means you can select some columns to export.  
  - Change of column order is supported, which means you can export the columns in an order different from the schema order of the table.  
  - Constant configuration is supported, and you need to follow the JSON format ["id", "1", "'bazhen.csy'", "null", "to_char(a + 1)", "2.3", "true"].
  - ID is normal column name  
  - 1 For plastic digital Constants  
  - 'Bazarn. CSY' is a String constant  
  - Null is a null pointer  
  - To_char (a + 1) is a function expression  
  - 2.3 is a floating point number  
  - True is a Boolean Value  
  - Column must contain the specified column set to be synchronized and it cannot be blank. | Yes     | N/A           |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| splitPk   | If you specify the splitPk when using RDBMS Reader to extract data, it means that you want to use the fields represented by splitPk for data sharding. Then, the DataX starts concurrent tasks to synchronize data, which greatly improves data synchronization efficiency.  
  · If you are using splitPk, we recommend that you use the tables primary keys because the primary keys are generally even and data hot spots are less prone to split data fragments.  
  · Currently, splitPk only supports data sharding for integer data types. Other types such as floating point, string, and date are not supported. If you specify an unsupported data type, DB2 Reader reports an error.  
  · If you do not fill in splitPk, you will be treated as if you do not split the single table, RDBMS reader uses a single channel to synchronize full data. | No | Blank |
| where     | The filtering condition. RDBMS Reader concatenates an SQL command based on specified column, table, and WHERE statements and extracts data according to the SQL. For example, you can specify the WHERE statement as limit 10 during a test. In actual business scenarios, the data on the current day is usually required for synchronization. You can specify the WHERE statement as gmt_create > $bizdate.  
  · The WHERE statement can be effectively used for incremental synchronization.  
  · If the WHERE statement is not set or is left null, full table data synchronization is applied. | No | N/A |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>querySql</td>
<td>In some business scenarios, the WHERE statement is insufficient for filtering. In such cases, the user can customize a filter SQL using this configuration item. When you configure this, the data synchronization system ignores the Table, column, and so on, filter the data directly using the contents of this configuration item. For example, you need to synchronize data after a multi-table join, using <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. When querySQL is configured, RDBMS Reader directly ignores the configuration of table, column, and WHERE statements.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>fetchSize</td>
<td>It defines the pieces of batch data that the plug-in and database server can fetch each time. The value determines the number of network interactions between the DataX system and the server, which can greatly improve data extraction performance. Note: The fetchsize value (&gt; 2048) may cause the data synchronization process Out of Memory (O).</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>

Development in wizard mode

Development in wizard mode is not supported currently.

Development in script mode

Configure a job to synchronously extract data from an RDBMS database:

```json
{
   "order": {
      "hops": [
         {
            "from": "Reader",
            "to": "Writer"
         }
      ]
   },
   "setting": {
      "errorLimit": {
         "record": "0"
      }
   }
}
```
{ "speed": {  "concurrent": 1,  "dmu": 1,  "throttle": false } },
"steps": [
{  "category": "reader",  "name": "Reader",  "parameter": {  "column": [  {   "type": "string",   "value": "field"  },  {   "type": "long",   "value": 100  },  {   "dateFormat": "yyyy-MM-dd HH:mm:ss",   "type": "date",   "value": "2014-12-12 12:12:12"  }  ],  "sliceRecordCount": "10"  },  "stepType": "stream" },
{  "category": "writer",  "name": "Writer",  "parameter": {  "connection": [  {   "jdbcUrl": "jdbc:dm://ip:port/database",   "table": [   "table"  ]  }  ],  "username": "username",  "password": "password",  "table": "table",  "column": [  "*"  ],  "preSql": [  "delete from XXX;"  ]  },  "stepType": "rdbms" }]
1.6.4.18 Configure Stream Reader

This topic describes data types and parameters supported by Stream Reader and how to configure Reader in script mode.

The Stream Reader plug-in provides the ability to automatically generate data from memory. It is mainly applicable to performance testing for data synchronization and basic functional testing.

The data types supported by stream reader are shown below.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Type description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Characters</td>
</tr>
<tr>
<td>long</td>
<td>Long Integer</td>
</tr>
<tr>
<td>date</td>
<td>Date type</td>
</tr>
<tr>
<td>bool</td>
<td>boolean</td>
</tr>
<tr>
<td>bytes</td>
<td>Bytes type</td>
</tr>
</tbody>
</table>
Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column data and type of generated source data. Multiple columns can be configured. You can set to generate random strings and specify the corresponding range. The example is as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;column&quot;: [</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;random&quot;: &quot;8, 15&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;random&quot;: &quot;10, 10&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configurations:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &quot;random&quot;: &quot;8,15&quot;:means to generate a random string with a length of 8-15 bytes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &quot;random&quot;: &quot;10,10&quot;:means to generate a random string with a length of 10 bytes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sliceRecordCount</td>
<td>Represents the number of copies that the loop generates column.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Development in wizard mode

Development in wizard mode is not supported currently.

Development in script mode

Configure a synchronization job to read data from memory:

```json
{
"type": "job",
"version": "1.0"}  //Indicates the version.
"steps": [
  {
    "stepType": "stream", //plug-in name
    "parameter": {
      "column": []  // Field
      {
        "type": "string", //Value Type
        "value": "field"  //Value
      },
      {
        "type": "long",
        "value": 100
      },
      {
```
"dateFormat": "yyyy-MM-dd HH:mm:ss", //time format
    "type": "date",
    "value": "2014-12-12 12:12:12"
  },
  {
    "type": "bool",
    "value": true
  },
  {
    "type": "bytes",
    "value": "byte string"
  }
],
"sliceRecordCount": "100000" //Represents the number of column generated by the loop.
},
"name": "Reader ",
"category": "reader"
}
{//The following is a writer template. You can find the corresponding writer plug-in documentations.
  "stepType": "stream ",
  "Parameter ":{}
  "name": "Writer",
  "category": "writer"
},
"setting": {
  "errorLimit": {
    "record": "0" //Number of error records
  },
  "speed": {
    "throttle": false, //false stands for open current, the speed of the lower limit does not work, and true stands for current limit
    "concurrent": "1", //Number of concurrent tasks
    "dmu": 1 //DMU Value
  }
},
"order":{
  "hops":[
  {
    "from": "Reader",
    "to": "Writer"
  }
  ]
}
}

1.6.4.19 Configure HybridDB for MySQL Reader

HybridDB for MySQL Reader can read tables and views. For table fields, you can specify all columns in sequence, specify certain columns, adjust the column order, specify constant fields, and configure HybridDB for MySQL functions such as now().

HybridDB for MySQL Reader connects to a remote HybridDB for MySQL database through a JDBC connector, generates SELECT SQL statements based on your
configuration, and sends the statements to the remote database. Then, HybridDB for MySQL Reader assembles SQL execution results into abstract datasets in custom data types of Data Integration, and passes the datasets to the downstream writer. At the same time, HybridDB for MySQL Reader runs a SELECT statement to read data from the HybridDB for MySQL database.

Type conversion list

HybridDB for MySQL Reader converts the data types in HybridDB for MySQL as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>HybridDB for MySQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int, Tinyint, Smallint, Mediumint, and Bigint</td>
</tr>
<tr>
<td>Float</td>
<td>Float, Double, and Decimal</td>
</tr>
<tr>
<td>String</td>
<td>Varchar, Char, Tinytext, Text, Mediumtext, and Longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date, Datetime, Timestamp, Time, and Year</td>
</tr>
<tr>
<td>Boolean</td>
<td>Bit and Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Tinyblob, Mediumblob, Blob, Longblob, and Varbinary</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data sources is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The name of the source table. A Data Integration job can synchronize only one table.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| column    | An array of columns to be synchronized from the configured table, in JSON format. The default value is [ * ], which indicates all columns.  
  - Column pruning is supported, which means that you can select and export specific columns.  
  - Change of column order is supported, which means that you can export the columns in an order different from the schema order of the table.  
  - Constant configuration is supported. Constants must be configured by using the SQL syntax, for example,  
    ```
    ["id","table","1","mingya.wmy","null","to_char(a+1)","2.3","true"]
    ```  
  - id: A common column name.  
  - table: A column name that contains a reserved word.  
  - 1: An integer constant.  
  - 'mingya.wmy': A string constant enclosed in single quotation marks.  
  - null: A null pointer.  
  - CHAR_LENGTH(s): A function used to calculate the string length.  
  - 2.3: A floating-point number.  
  - true: A Boolean value.  
  - The column attribute must explicitly specify a set of columns to be synchronized. It cannot be left blank. | Yes       | None          |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
</table>
| splitPk   | The field used for data sharding when HybridDB for MySQL Reader extracts data. If you specify splitPk, Data Integration initiates concurrent tasks to synchronize data, which greatly improves the efficiency of data synchronization.  
  - We recommend that you set the splitPk attribute to the primary key of the table. Based on primary keys, data can be well distributed to different shards, but not intensively distributed to certain shards.  
  - Currently, the splitPk attribute supports data sharding only for integers but not for other data types such as String, Float, and Date. If you specify an unsupported data type, Data Integration ignores the splitPk attribute and synchronizes data through a single task.  
  - If you do not provide the splitPk attribute or leave it blank, Data Integration synchronizes the table data through a single task. | No       | None          |
| where     | The filter condition. In actual business scenarios, the data on the current day is usually synchronized. In this case, you can set the where attribute to gmt_create>$bizdate.  
  - The where attribute can be used to synchronize incremental business data effectively. If the where attribute is not specified (for example, the key or value of the where attribute is not provided), full synchronization is performed.  
  - You cannot set the where attribute to limit 10, which does not conform to the constraints of HybridDB for MySQL on the SQL WHERE clause.                                                                 | No       | None          |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>querySql (an advanced attribute, which is not available in wizard mode)</td>
<td>The custom filter SQL statement used in some business scenarios where the filter condition specified by the where attribute is insufficient. After this attribute is set, Data Integration ignores the table, column, and splitPk attributes, but directly filters data based on this attribute. For example, to synchronize data after joining multiple tables, set the querySql attribute to select a,b from table_a join table_b on table_a.id = table_b.id. The priority of querySql is higher than those of table, column, where, and splitPk. When querySql is set, HybridDB for MySQL Reader directly ignores the configuration of the table, column, where, or splitPk attribute. The data source uses querySql to parse out information such as the username and password.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>singleOrMulti (applicable only to database and table sharding)</td>
<td>Indicates whether to perform database and table sharding. When you switch from the wizard mode to script mode, the following configuration is automatically generated: &quot;singleOrMulti&quot;: &quot;multi&quot;. However, the task script configuration template does not automatically generate this configuration. You need to add it manually. Otherwise, only the first data source is recognized. The singleOrMulti attribute is used only at the front end. The back end does not use this attribute to determine whether to perform database and table sharding.</td>
<td>Yes</td>
<td>multi</td>
</tr>
</tbody>
</table>

**Development in wizard mode**

Configure the source and destination of data for a synchronization task.
### Parameter | Description
--- | ---
**Data Source** | The datasource attribute in the preceding parameter description. Select the data source that you have configured.
**Table** | The table attribute in the preceding parameter description. Select the source table.
**Filter** | The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. SQL syntaxes vary with data sources.
**Shard Key** | The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.

### Note:
The Shard Key parameter is displayed only when you configure the source of data for a synchronization task.

Configure mappings of fields (the column attribute in the preceding parameter description).

Each source table field on the left maps a destination table field on the right. You can click Add to add a mapping or move the cursor over a line and click Delete to delete the current mapping.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Fields with the Same Name</td>
<td>Click Map Fields with the Same Name to establish a mapping between fields with the same name. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Map Fields in the Same Line</td>
<td>Click Map Fields in the Same Line to establish a mapping for the same row. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Remove Mappings</td>
<td>Click Remove Mappings to remove mappings that have been established.</td>
</tr>
<tr>
<td>Auto Layout</td>
<td>The fields are automatically sorted based on specified rules.</td>
</tr>
<tr>
<td>Change Fields in Source Table</td>
<td>You can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, while other blank rows are ignored.</td>
</tr>
<tr>
<td>Add</td>
<td>Click Add to add a mapping.</td>
</tr>
<tr>
<td></td>
<td>• You can enter constants. Each constant must be enclosed in single quotation marks, such as 'abc' and '123'.</td>
</tr>
<tr>
<td></td>
<td>• You can use scheduling parameters, such as ${bizdate}.</td>
</tr>
<tr>
<td></td>
<td>• You can enter functions supported by relational databases, such as now() and count(1).</td>
</tr>
<tr>
<td></td>
<td>• If the value you entered cannot be parsed, the type is displayed as Unidentified.</td>
</tr>
</tbody>
</table>

Configure channel control
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent Jobs</td>
<td>The maximum number of threads used to concurrently read data from the source or write data into the data storage media in a data synchronization task. In wizard mode, you can configure the concurrency for a task on the wizard page.</td>
</tr>
<tr>
<td>Dirty Data Records Allowed</td>
<td>The maximum number of errors or dirty data records allowed.</td>
</tr>
<tr>
<td>Task Resource Group</td>
<td>The machines on which tasks are run. If a large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group. For more information, see &lt;xref&gt;#unique_12</td>
</tr>
</tbody>
</table>

Development in script mode

The following code is an example of configuration for a single table in one database. For more information about attributes, see the preceding parameter description.

```json
{
    "type": "job",
    "steps": [
        {
            "parameter": {
                "datasource": "px_aliyun_hymysql",// The data source name.
                "column": [ // The source table columns.
                    "id",
                    "name",
                    "sex",
                    "salary",
                    "age",
                    "pt"
                ],
                "where": "id=10001";// The filter condition.
                "splitPk": "id";// The shard key.
                "table": "person"// The source table name.
            },
            "name": "Reader",
            "category": "reader"
        },
        {
```
1.6.4.20 Configure AnalyticDB for PostgreSQL Reader

This topic describes the data types and parameters supported by AnalyticDB for PostgreSQL Reader and how to configure it in both wizard and script modes.

AnalyticDB for PostgreSQL Reader reads data from a AnalyticDB for PostgreSQL database. At the underlying implementation level, AnalyticDB for PostgreSQL Reader connects to a remote AnalyticDB for PostgreSQL database through JDBC and runs SELECT statements to extract data from the database. On the public cloud, RDS provides the AnalyticDB for PostgreSQL storage engine.

In short, AnalyticDB for PostgreSQL Reader connects to a remote AnalyticDB for PostgreSQL database through a JDBC connector, generates SELECT statements based on your configuration, and sends the statements to the remote database. Then, AnalyticDB for PostgreSQL Reader assembles SQL execution results into abstract datasets in custom data types of Data Integration, and passes the datasets to the downstream writer.

- AnalyticDB for PostgreSQL Reader concatenates the configured table, column, and where information into SQL statements and sends the statements to the AnalyticDB for PostgreSQL database.
- AnalyticDB for PostgreSQL Reader sends the configured querySql information directly to the AnalyticDB for PostgreSQL database.
DataWorks

Data Aggregation / 1 Workbench

Type conversion list

AnalyticDB for PostgreSQL Reader supports most data types in AnalyticDB for PostgreSQL. Check whether a data type is supported before configuring AnalyticDB for PostgreSQL Reader.

AnalyticDB for PostgreSQL Reader converts the data types in AnalyticDB for PostgreSQL as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>AnalyticDB for PostgreSQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Bigint, Bigserial, Integer, Smallint, and Serial</td>
</tr>
<tr>
<td>Float</td>
<td>Double precision, Money, Numeric, and Real</td>
</tr>
<tr>
<td>String</td>
<td>Varchar, Char, Text, Bit, and Inet</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date, Time, and Timestamp</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Bytea</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data sources is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The name of the source table.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>An array of columns to be synchronized from the configured table, in JSON format. The default value is [ * ], which indicates all columns.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>

- Column pruning is supported, which means that you can select and export specific columns.
- Change of column order is supported, which means that you can export the columns in an order different from the schema order of the table.
- Constant configuration is supported. Constants must be configured by using the SQL syntax, for example, `["id", "table","1", "'mingya.wmy'", "'null'", "to_char(a+1)", "2.3", "true"]`.  
  - id: A common column name.
  - table: A column name that contains a reserved word.
  - 1: An integer constant.
  - 'mingya.wmy': A string constant enclosed in single quotation marks.
  - null: A null pointer.
  - CHAR_LENGTH(s): A function used to calculate the string length.
  - 2.3: A floating-point number.
  - true: A Boolean value.
- The column attribute must explicitly specify a set of columns to be synchronized. It cannot be left blank.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
</table>
| splitPk   | The field used for data sharding when AnalyticDB for PostgreSQL Reader extracts data. If you specify splitPk, Data Integration initiates concurrent tasks to synchronize data, which greatly improves the efficiency of data synchronization.  
  · We recommend that you set the splitPk attribute to the primary key of the table. Based on primary keys, data can be well distributed to different shards, but not intensively distributed to certain shards.  
  · Currently, the splitPk attribute supports data sharding only for integers but not for other data types such as String, Float, and Date. If you specify an unsupported data type, Data Integration ignores the splitPk attribute and synchronizes data through a single task.  
  · If you do not provide the splitPk attribute or leave it blank, Data Integration synchronizes the table data through a single task. | No | None |
| where     | The filter condition. AnalyticDB for PostgreSQL Reader concatenates the specified column, table, and where information into an SQL statement and uses the SQL statement to extract data. For example, you can set the where attribute to id>2 and sex=1 during a test. In actual business scenarios, the data on the current day is usually synchronized.  
  · The where attribute can be used to synchronize incremental business data effectively.  
  · If you do not provide the where attribute or leave it blank, the data of the entire table is synchronized. | No | None |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>querySql (an advanced attribute, which is not available in wizard mode)</td>
<td>The custom filter SQL statement used in some business scenarios where the filter condition specified by the where attribute is insufficient. After this attribute is set, Data Integration ignores the table, column, and splitPk attributes, but directly filters data based on this attribute. For example, to synchronize data after joining multiple tables, set the querySql attribute to <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. [DO NOT TRANSLATE]</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>fetchSize</td>
<td>The number of data records that the plug-in can fetch from the database server each time. The value determines the frequency of interaction between Data Integration and the server on the network, and therefore can be used to greatly improve data extraction performance.</td>
<td>No</td>
<td>512</td>
</tr>
</tbody>
</table>

Note:
A value greater than 2048 may lead to OOM during the data synchronization process.
Development in wizard mode

1. Specify data sources.

Configure the source and destination of data for a synchronization task.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>The datasource attribute in the preceding parameter description. Select the data source that you have configured.</td>
</tr>
<tr>
<td>Table</td>
<td>The table attribute in the preceding parameter description. Select the source table.</td>
</tr>
<tr>
<td>Filter</td>
<td>The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. SQL syntaxes vary with data sources.</td>
</tr>
<tr>
<td>Shard Key</td>
<td>The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.</td>
</tr>
</tbody>
</table>

Note:
The Shard Key parameter is displayed only when you configure the source of data for a synchronization task.
2. Configure mappings of fields (the column attribute in the preceding parameter description).

Each source table field on the left maps a destination table field on the right. You can click Add to add a mapping or move the cursor over a line and click Delete to delete the current mapping.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Fields with the Same Name</td>
<td>Click Map Fields with the Same Name to establish a mapping between fields with the same name. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Map Fields in the Same Line</td>
<td>Click Map Fields in the Same Line to establish a mapping for the same row. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Remove Mappings</td>
<td>Click Remove Mappings to remove mappings that have been established.</td>
</tr>
<tr>
<td>Auto Layout</td>
<td>The fields are automatically sorted based on specified rules.</td>
</tr>
<tr>
<td>Change Fields in Source Table</td>
<td>You can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, while other blank rows are ignored.</td>
</tr>
</tbody>
</table>
| Add                         | • You can enter constants. Each constant must be enclosed in single quotation marks, such as 'abc' and '123'.  
                               | • You can use scheduling parameters, such as ${bizdate}.
                               | • You can enter functions supported by relational databases, such as now() and count(1). 
                               | • If the value you entered cannot be parsed, the type is displayed as Unidentified. |
3. Configure channel control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent Jobs</td>
<td>The maximum number of threads used to concurrently read data from the source or write data into the data storage media in a data synchronization task. In wizard mode, you can configure the concurrency for a task on the wizard page.</td>
</tr>
<tr>
<td>Dirty Data Records</td>
<td>The maximum number of errors or dirty data records allowed.</td>
</tr>
<tr>
<td>Task Resource Group</td>
<td>The machines on which tasks are run. If a large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group. Currently, a custom resource group can be added only in China (Hangzhou) and China (Shanghai). For more information, see #unique_12.</td>
</tr>
</tbody>
</table>

Development in script mode

```json
{
    "type": "job",
    "steps": [
        {
            "parameter": {
                "datasource": "test_004", // The data source name.
                "column": [ // The source table columns.
                    "id",
                    "name",
                    "sex",
                    "salary",
                    "age",
                ],
                "where": "id=1001", // The filter condition.
                "splitPk": "id", // The shard key.
                "table": "public.person" // The source table name.
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "parameter": {},
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0", // The version number.
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```
1.6.4.21 Configure POLARDB Reader

This topic describes the data types and parameters supported by POLARDB Reader and how to configure it in both the wizard and script modes.

POLARDB Reader connects to a remote POLARDB database through a JDBC connector, generates SELECT statements based on your configuration, and sends the statements to the remote database. Then, POLARDB Reader assembles SQL execution results into abstract datasets in custom data types of Data Integration, and passes the datasets to the downstream writer.

In short, POLARDB Reader connects to a remote POLARDB database through a JDBC connector and runs SELECT statements to extract data from the remote database at the underlying layer. POLARDB Reader can read tables and views. For table fields, you can specify all or some of the columns in sequence, adjust the column order, specify constant fields, and configure POLARDB functions, such as now().

Type conversion list

POLARDB Reader converts the data types in POLARDB as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>POLARDB data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int, Tinyint, Smallint, Mediumint, and Bigint</td>
</tr>
<tr>
<td>Float</td>
<td>Float, Double, and Decimal</td>
</tr>
<tr>
<td>String</td>
<td>Varchar, Char, Tinytext, Text, Mediumtext, and Longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date, Datetime, Timestamp, Time, and Year</td>
</tr>
<tr>
<td>Boolean</td>
<td>Bit and Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Tinyblob, Mediumblob, Blob, Longblob, and Varbinary</td>
</tr>
</tbody>
</table>

Note:

- Except the preceding field types, other types are not supported.
- POLARDB Reader classifies tinyint(1) as the integer type.

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data sources is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The name of the source table. A Data Integration job can synchronize only one table.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>An array of columns to be synchronized from the configured table, in JSON format. The default value is [ * ], which indicates all columns.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>

- Column pruning is supported, which means that you can select and export specific columns.
- Change of column order is supported, which means that you can export the columns in an order different from the schema order of the table.
- Constant configuration is supported. Constants must be configured by using the SQL syntax, for example, `["id", "table","1", "'mingya.wmy'", "'null'", "to_char(a+1)", "2.3", "true"]`.

  - id: A common column name.
  - table: A column name that contains a reserved word.
  - 1: An integer constant.
  - 'mingya.wmy': A string constant enclosed in single quotation marks.
  - null: A null pointer.
  - CHAR_LENGTH(s): A function used to calculate the string length.
  - 2.3: A floating-point number.
  - true: A Boolean value.

- The column attribute must explicitly specify a set of columns to be synchronized. It cannot be left blank.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
</table>
| splitPk   | The field used for data sharding when POLARDB Reader extracts data. If you specify splitPk, Data Integration initiates concurrent tasks to synchronize data, which greatly improves the efficiency of data synchronization.  
  · We recommend that you set the splitPk attribute to the primary key of the table. Based on primary keys, data can be well distributed to different shards, but not intensively distributed to certain shards.  
  · Currently, the splitPk attribute supports data sharding only for integers but not for other data types such as String, Float, and Date. If you specify an unsupported data type, Data Integration ignores the splitPk attribute and synchronizes data through a single task.  
  · If you do not provide the splitPk attribute or leave it blank, Data Integration synchronizes the table data through a single task. | No | None |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
</table>
| where                        | The filter condition. In actual business scenarios, the data on the current day is usually synchronized. In this case, you can set the where attribute to gmt_create>$bizdate.  
  · The where attribute can be used to synchronize incremental business data effectively. If the where attribute is not specified (for example, the key or value of the where attribute is not provided), full synchronization is performed.  
  · You cannot set the where attribute to limit 10, which does not conform to the constraints on the SQL WHERE clause. | No       | None          |
| querySql (an advanced attribute, which is not available in wizard mode) | The custom filter SQL statement used in some business scenarios where the filter condition specified by the where attribute is insufficient. After this attribute is set, Data Integration ignores the table, column, and splitPk attributes, but directly filters data based on this attribute. For example, to synchronize data after joining multiple tables, set the querySql attribute to `select a,b from table_a join table_b on table_a.id = table_b.id`. The priority of querySql is higher than those of table, column, where, and splitPk. When querySql is set, POLARDB Reader directly ignores the configuration of the table, column, where, or splitPk attribute. The data source uses querySql to parse out information such as the username and password. | No       | None          |
### Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>singleOrMulti (applicable only to database and table sharding)</td>
<td>Indicates whether to perform database and table sharding. When you switch from the wizard mode to script mode, the following configuration is automatically generated: &quot;singleOrMulti&quot;: &quot;multi&quot;. However, the task script configuration template does not automatically generate this configuration. You need to add it manually. Otherwise, only the first data source is recognized. The singleOrMulti attribute is used only at the front end. The back end does not use this attribute to determine whether to perform database and table sharding.</td>
<td>Yes</td>
<td>multi</td>
</tr>
</tbody>
</table>

Development in wizard mode

1. Specify data sources.

Configure the source and destination of data for a synchronization task.

![Data Source Configuration](image)

### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>The datasource attribute in the preceding parameter description. Select the data source that you have configured.</td>
</tr>
<tr>
<td>Table</td>
<td>The table attribute in the preceding parameter description. Select the source table.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Statements Run Before Import</td>
<td>The preSql attribute in the preceding parameter description. Enter the SQL statement that is run before the data synchronization task is run.</td>
</tr>
<tr>
<td>Statements Run After Import</td>
<td>The postSql attribute in the preceding parameter description. Enter the SQL statement that is run after the data synchronization task is run.</td>
</tr>
<tr>
<td>Primary Key Violation</td>
<td>The writeMode attribute in the preceding parameter description. Select the expected write mode.</td>
</tr>
</tbody>
</table>

2. Configure mappings of fields (the column attribute in the preceding parameter description).

Each source table field on the left maps a destination table field on the right. You can click Add to add a mapping or move the cursor over a line and click Delete to delete the current mapping.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Fields with the Same Name</td>
<td>Click Map Fields with the Same Name to establish a mapping between fields with the same name. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Map Fields in the Same Line</td>
<td>Click Map Fields in the Same Line to establish a mapping for the same row. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Remove Mappings</td>
<td>Click Remove Mappings to remove mappings that have been established.</td>
</tr>
<tr>
<td>Auto Layout</td>
<td>The fields are automatically sorted based on specified rules.</td>
</tr>
</tbody>
</table>
### 3. Configure channel control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMU</td>
<td>The unit that measures the resources (including CPU, memory, and network resources) consumed by Data Integration. A DMU represents the minimum operating capability of a Data Integration task, that is, the data synchronization processing capability given limited CPU, memory, and network resources.</td>
</tr>
<tr>
<td>Concurrent Jobs</td>
<td>The maximum number of threads used to concurrently read data from the source or write data into the data storage media in a data synchronization task. In wizard mode, you can configure the concurrency for a task on the wizard page.</td>
</tr>
<tr>
<td>Dirty Data Records Allowed</td>
<td>The maximum number of errors or dirty data records allowed.</td>
</tr>
<tr>
<td>Task Resource Group</td>
<td>The machines on which tasks are run. If a large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group. Currently, a custom resource group can be added only in China (Hangzhou) and China (Shanghai). For more information, see Add task resources.</td>
</tr>
</tbody>
</table>

### Development in script mode

The following code is an example of configuration for a single table in one database. For more information about attributes, see the preceding parameter description.

```json
{
    "type": "job",
    "steps": [
        {
            "parameter": {
                "datasource": "test_005", // The data source name.
                "column": [ // The source table columns.
                    "id",
                    "name",
                    ... // Additional columns
                ]
            }
        }
    ]
}
```
1.6.5 Configure writer plug-in

1.6.5.1 Configure AnalyticDB for MySQL 2.0 Writer

This topic describes the data types and parameters supported by AnalyticDB for MySQL 2.0 Writer and how to configure Writer in both wizard and script mode.

AnalyticDB for MySQL 2.0 Writer allows you to write data to AnalyticDB for MySQL 2.0 in the following two modes:

- Load Data (batch import): Transfers and loads data from the data source to AnalyticDB for MySQL 2.0.
  - Advantage: Imports a large volume of data (more than 10 million data records) at a high speed.
  - Disadvantage: Authorization from the third party is required.
- Insert Ignore (real-time insertion): Directly writes data to AnalyticDB for MySQL 2.0.
  - Advantage: Writes a small volume of data (less than 10 million data records) at a high speed.
  - Disadvantage: Unsuitable for writing a large volume of data due to a low speed.

You must configure the data source before configuring the AnalyticDB for MySQL 2.0 Writer plug-in. For more information, see Configure AnalyticDB for MySQL 2.0 data source.

AnalyticDB for MySQL 2.0 Writer supports the following data types in AnalyticDB for MySQL 2.0:

<table>
<thead>
<tr>
<th>Type</th>
<th>AnalyticDB for MySQL 2.0 data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>int, tinyint, smallint, int, bigint</td>
</tr>
<tr>
<td>Floating point</td>
<td>float and double</td>
</tr>
<tr>
<td>String type</td>
<td>varchar</td>
</tr>
<tr>
<td>Date and time</td>
<td>date</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
</tr>
</tbody>
</table>

Prerequisites

- Before importing data in Load Data mode with a MaxCompute table as the data source, you must grant the Describe and Select permissions for the table to the import account of AnalyticDB for MySQL 2.0 in MaxCompute.

Public cloud accounts are garuda_build@aliyun.com and garuda_data@aliyun.com. Authorization is required for both accounts. For the import accounts of private clouds, see the configuration documents of relevant private clouds. Generally, the import account of a private cloud is test1000000009@aliyun.com.

Command for granting permissions:

```sql
USE projectname;--The MaxCompute project to which the table belongs.
ADD USER ALIYUN$xxxx@aliyun.com;--Enter a correct cloud account (when adding the account for the first time).
```
GRANT Describe, Select ON TABLE table_name TO USER ALIYUN$xxxx@aliyun.com; -- Enter the table on which permissions are granted and a correct cloud account.

To ensure your data security, only the data from the MaxCompute Project in which the operator is the project owner or MaxCompute table owner can be imported to AnalyticDB for MySQL 2.0. Most of private clouds have no such restriction.

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>AnalyticDB for MySQL 2.0 connection information in the form ip:port.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>schema</td>
<td>The schema name of the AnalyticDB for MySQL 2.0.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>username</td>
<td>The user name of the AnalyticDB for MySQL 2.0 account, which is the current AccessID.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>password</td>
<td>The password of the AnalyticDB for MySQL 2.0 account, which is the current AccessKey.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>datasource</td>
<td>The data source name. The name entered here must be the same as the added data source. You can add a data source in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The name of the target table.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| partition | The partition name of the target table. If the target table is partitioned, this field is required. If the Reader is MaxCompute, and AnalyticDB for MySQL 2.0 Writer imports data in Load Data mode, the partitions of MaxCompute only support the following three configurations (take two-level partitions as an example):
  - "partition" : ["pt=\*, ds=\*\*"] (reads data from all partitions under the table)
  - "partition" : ["pt=1, ds=\*\*"] (reads data from all the secondary partitions under the primary partition pt=1 under the table)
  - "partition" : ["pt=1, ds=hangzhou"] (reads data from the secondary partition ds=hangzhou under the primary partition pt=1 under the table) | No       | None          |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>writeMode</td>
<td>Insert mode. If the record with the same primary key already exists, the old record is discarded.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>column</td>
<td>The list of fields in the target table. The value can be [&quot;*&quot;] or a list of specific fields, such as [&quot;a&quot;, &quot;b&quot;, &quot;c&quot;].</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>overWrite</td>
<td>Specified whether to overwrite the current target table when writing data to AnalyticDB for MySQL 2.0. True means the table is overwritten, and False means that the table is not overwritten and the data is appended. This value takes effect only if the writeMode is Load.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>lifeCycle</td>
<td>The life cycle of an AnalyticDB for MySQL 2.0 temporary table. This value takes effect only if the writeMode is Load.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>suffix</td>
<td>The AnalyticDB for MySQL 2.0 URL is in the format of ip:port, which changes to a JDBC database connection string upon access to AnalyticDB for MySQL 2.0. This parameter is a custom connection string and is optional. See the JDBC control parameters supported by MySQL. For example, configure the suffix to autoReconnect=true&amp;failOverReadOnly=false&amp;maxReconnects=10. Required: No</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>opIndex</td>
<td>Subscript of the Operation Type column of AnalyticDB for MySQL 2.0 peer storage, which starts from 0. This value takes effect only if the writeMode is stream.</td>
<td>Required</td>
<td>N/A</td>
</tr>
<tr>
<td>batchSize</td>
<td>Number of data items of each batch committed to AnalyticDB for MySQL 2.0. This value takes effect only if the writeMode is Insert.</td>
<td>Required</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bufferSize</td>
<td>Size of the DataX data buffer. The buffers are aggregated to form a large buffer. The data from the source is collected to this buffer for sorting before being committed to AnalyticDB for MySQL 2.0. The data is sorted by the AnalyticDB for MySQL 2.0 partition column so that data is organized in an order that is more friendly for the AnalyticDB for MySQL 2.0 server to improve the performance. The data in the buffer with a size of BufferSize is committed to AnalyticDB for MySQL 2.0 in batches with a size of batchSize. The bufferSize value must be set to a multiple of batchSize. This value takes effect only if the writeMode is insert.</td>
<td>Required</td>
<td>Default value: This feature is disabled by default.</td>
</tr>
</tbody>
</table>

Introduction to wizard mode

1. Choose source

Configure the source and destination of the data for the synchronization task.

Configuration item descriptions:

- Data source: The datasource in the preceding parameter description. Enter the configured data source name.
- Table: The table in the preceding parameter description. Select the table for synchronization.
- Import mode: The writeMode in the preceding parameter description. Load Data (batch import) and Insert Ignore (real-time insertion) modes are supported.
2. Field mapping: The column in the preceding parameter description.

The source table field on the left and the target table field on the right are one-to-one relationships, click Add row to add a single field and click Delete to delete the current field.

- Peer mapping: Click peer mapping to establish a corresponding mapping relationship in the peer that matches the data type.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Channel control

Configurations:

- DMU: A unit that measures the resources consumed during data integration, including CPU, memory, and network bandwidth. It represents a unit of data synchronization processing capability given limited CPU, memory, and network resources.
- Concurrent count: The maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- Number of error records: The maximum number of dirty data records.
Development in script mode

```json
{
   "type":"job",
   "version":"2.0",
   "steps": [ // below is the template for reader, you can find the appropriate read plug-in documentation.
     {
       "stepType":"stream",
       "parameter":{
         "name":"Reader",
         "category":"reader"
       },
       { // plug-in name
         "stepType":"ads",
         "parameter":{
           "partition": "", // partition name of the target table
           "datasource": "", // Data Source
           "column": [ // Field
             "id"
           ],
           "writeMode": "insert", // Write mode
           "batchSize": "1000", // number of records submitted in one batch size
           "table": "", // The name of the target table.
           "overWrite": "true" // AnalyticDB for MySQL 2.0 write whether or not to override the currently written table, true is an overlay write, and false is a non-override (append) Write. This value takes effect only if the writeMode is Load.
         },
         "name": "Writer",
         "category": "writer"
       }
     ],
     "setting": {
       "errorLimit": {
         "record": "0" // Number of error records
       },
       "speed": {
         "throttle": false, // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
         "concurrent": "1", // Number of concurrent tasks
       }
     },
     "order": {
       "hops": [
         {
           "from": "Reader",
           "to": "Writer"
         }
       ]
     }
   }
}```
1.6.5.2 Configure DataHub Writer
This topic describes the data types and parameters supported by DataHub Writer and how to configure Writer in script mode.

DataHub is a real-time data distribution and streaming data processing platform. It can publish, subscribe, and distribute streaming data. It allows you to easily create analysis programs and applications based on streaming data.

Based on Alibaba Cloud's Apsara platform, DataHub delivers high availability, low latency, high scalability, and high throughput. Seamlessly connect to Alibaba Cloud's stream computing engine, StreamCompute, DataHub allows you to easily use SQL statements to analyze streaming data. DataHub provides the function to distribute streaming data to cloud products, currently including MaxComputer and Object Storage System (OSS).

Note:
The string can only be UTF-8 encoded and the maximum length of a single string column is 1 MB.

Parameter configuration

The source is connected to the sink through a channel. The channel type at the writer must be consistent with that at the Reader. Two types of channels are provided generally: memory channel and file channel. The following example describes how to configure a file channel.

"agent.sinks.dataXSinkWrapper.channel": "file"

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessId</td>
<td>The accessId of the Datahub.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>accessKey</td>
<td>The accessKey of the DataHub.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>endpoint</td>
<td>For an access request to a DataHub resource, select the correct domain name based on the service that the resource belongs.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>maxRetryCount</td>
<td>The maximum number of retries for task failure.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>mode</td>
<td>The write mode when the value type is string.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>parseContent</td>
<td>Parses the content.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>project</td>
<td>Project is the basic unit of DataHub data that contains multiple topics.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>topic</td>
<td>Topic is the smallest unit of the DataHub subscription and publication, you can use topic to represent one type or one type of streaming data.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>maxCommitSize</td>
<td>To improve writing efficiency, DataX-On-Flume collects the buffer data and submits it to the target end in batches when the collected data size reaches maxCommitSize (in MB). The maxCommitSize is 1 MB by default.</td>
<td>No</td>
<td>1 MB</td>
</tr>
<tr>
<td>batchSize</td>
<td>To improve writing efficiency, DataX-On-Flume collects the buffer data and submits it to the target end in batches when the number of collected data entries reaches batchSize (in entry). The batchSize is 1024 entries by default.</td>
<td>No</td>
<td>1,024</td>
</tr>
<tr>
<td>maxCommitInterval</td>
<td>To improve writing efficiency, DataX-On-Flume collects buffer data and submits it to the target end in batches when the number of collected data entries reaches the limit of maxCommitSize and batchSize. If the data collection source does not produce data for extensive periods, the maxCommitInterval parameter (the maximum time allowed for the buffer data preservation, beyond which the data is compulsively delivered in milliseconds) is increased to ensure the timely delivery of data. The maxCommitInterval is 30000 (30 seconds) by default.</td>
<td>No</td>
<td>30</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>parseMode</td>
<td>Log parsing mode includes non-parsing default mode and CSV mode. In the non-parsing mode, one collected log line is written directly as a column of DataX Record. The CSV mode supports configuring one column separator, which separates one log line into multiple columns of DataX Record.</td>
<td>No</td>
<td>default</td>
</tr>
</tbody>
</table>

**Development in wizard mode**

**Development in wizard mode is not supported currently.**

**Development in script mode**

**Configure a synchronization job to read data from memory:**

```json
{
   "type": "job",
   "version": "2.0", // version size
   "steps": [
   {//The following is a reader template. You can find the corresponding reader plug-in documentations.
    "stepType": "stream",
    "parameter": {},
    "name": "Reader",
    "category": "reader"
   },
   {
    "stepType": "datahub", // plug-in name
    "parameter": {
     "datasource": "", // Name of the data source
     "topic": "", // Topic is the smallest unit of DataHub subscription and publishing. You can use Topic to represent a class or a kind of streaming data.
     "maxRetryCount": 500, // Number of retries
     "maxCommitSize": 1048576 // data to be saved to buffer size reaches maxrefersizer size (in MB) when, batch submitted to the destination
    },
    "name": "Writer",
    "category": "writer"
   }
   ],
   "setting": {
    "errorLimit": {
     "record": "" // Number of error records
    },
    "speed": {
     "concurrent": 20, // Number of concurrent jobs
     "throttle": false, // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
     "dmu": 20 // DMU values
    }
   }
}```
1.6.5.3 Configure DB2 Writer

This topic describes the data types and parameters supported by DB2 Writer and how to configure Writer in script mode.

The DB2 Writer plug-in can write data into the target tables of DB2 databases. At the underlying implementation level, DB2 Writer connects to a remote DB2 database through JDBC, and runs the `insert into ...` SQL statement to write data into DB2. The data is submitted and written into the database in batches in DB2.

DB2 Writer is designed for ETL developers to import data from data warehouses to DB2. The DB2 Writer can also be used as a data migration tool by DBA and other users.

DB2 Writer acquires the protocol data generated by Reader by means of the Data Integration framework. When the `insert into...` SQL statement is run, if the primary key conflicts with the unique index, data cannot be written into the conflicting lines. To improve performance, we use `PreparedStatement + Batch` and configure `rewriteBatchedStatements=true` to buffer data to the thread context buffer. A write request is submitted only when the amount of data in the buffer reaches the threshold.

**Note:**
The task should at least have the `insert into...` permission. Whether other permissions are required depends on the statements specified in PreSQL and PostSQL when you configure the task.

DB2 Writer supports most data types in DB2. Check whether your data type is supported.

DB2 Writer converts DB2 data types as follows:
<table>
<thead>
<tr>
<th>Category</th>
<th>DB2 data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>Float</td>
<td>Decimal, real, and double</td>
</tr>
<tr>
<td>String</td>
<td>char, character, varchar, graphic, vargraphic, long varchar, clob, long vargraphic, or dbclob</td>
</tr>
<tr>
<td>Date and time type</td>
<td>decimal, real, and double</td>
</tr>
<tr>
<td>Boolean</td>
<td>—</td>
</tr>
<tr>
<td>Binary</td>
<td>blob</td>
</tr>
</tbody>
</table>

### Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdbcUrl</td>
<td>Information of the JDBC connection to the DB2 database. According to the DB2 official specification, jdbcUrl in the DB2 format is jdbc:db2://ip:port/database, and the URL attachment control information can be entered.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>username</td>
<td>The user name of the data source.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>password</td>
<td>Password corresponding to the specified user name for the data source.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The table selected for synchronization.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>column</td>
<td>The fields of the target table into which data is required to be written. These fields are separated by commas (,). For example: &quot;column&quot;: [&quot;id&quot;, &quot;name&quot;, &quot;age&quot;]. Use if it is required to write data into all columns in sequence. For example: &quot;column&quot;: [&quot;*&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement that is run before the data synchronization task is run. Currently, you can run only one SQL statement, for example, clear old data.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement that is run after the data synchronization task is run. Currently, you can run only one SQL statement in wizard mode, and more than one SQL statement in script mode. For example, add a timestamp.</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Attribute Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>batchSize</td>
<td>The quantity of records submitted in batches at a time. This parameter can greatly reduce the interactions between Data Integration and DB2 over the network, and increase the overall throughout. However, the running process of Data Integration may become out of memory (OOM) if the value is too large.</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>

**Development in wizard mode**

**Development in wizard mode is not supported currently.**

**Development in script mode**

**Configure the data synchronization job to write data to DB2:**

```json
{
    "type":"job",
    "version":2.0 , // version number
    "steps":[
        { //The following is a reader template. You can find the corresponding reader plug-in documentations.
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"db2", // plug-in name
            "parameter":{
                "postSql":[] , // SQL statement that was first executed before the data synchronization task was executed
                "password":"", // Password
                "jdbcUrl":"jdbc:db2://ip:port/database" ,//JDBC connection information for DB2 database
                "column":[
                    "id",
                ],
                "batchSize":1024 ,// number of records submitted in one batch size
                "table":"",
                "username":"",
                "preSql": []// SQL statement executed after the data synchronization task is executed
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" //Number of error records
        },
        "speed":{
```
1.6.5.4 Configure DRDS Writer

This topic describes the data types and parameters supported by DRDS Writer and how to configure Writer in both wizard and script mode.

The DRDS Writer plug-in provides the ability to write data to DRDS tables. At the underlying implementation level, the DRDS Writer connects to the proxy of a remote DRDS database through JDBC, and writes data into DRDS by running the corresponding SQL statement `replace into...`. The SQL statement writes the data to the DRDS.

**Note:**

Note that the SQL statement you run is `replace into`, and your table must have a primary key or a unique index to avoid data duplication. You must configure the data source before configuring the DRDS Writer plug-in. For more information, see [Configure DRDS data sources](#).

DRDS Writer is designed for ETL developers to import data from data warehouses to DRDS. DRDS Writer can also be used as a data migration tool by DBA and other users.

DRDS Writer acquires the protocol data generated by Reader by means of the CDP framework, and writes data into DRDS by running the statement `replace into...`. If the primary key does not conflict with the unique index, the system performs the same action with `insert into`. When a conflict exists, all the fields in the original line are replaced with the fields in the new line. DRDS Writer commits the accumulated data to DRDS's proxy, which then determines whether the data is written into one table or multiple tables, and how to route the data when it is written into multiple tables.
Note:
The entire task should at least have the permission replace into.... Whether other permissions are required depends on the statements you specified in PreSQL and PostSQL when you configure the task.

Similar to MySQL Writer, the DRDS Writer currently supports most data types in MySQL. Check whether your data type is supported.

DRDS Writer converts DRDS data types as follows:

<table>
<thead>
<tr>
<th>Type Classification</th>
<th>DRDS data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>int, tinyint, smallint, mediumint, int, bigint, and year</td>
</tr>
<tr>
<td>Floating point</td>
<td>float, double, and decimal</td>
</tr>
<tr>
<td>String</td>
<td>varchar, char, tinytext, text, mediumtext, and longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>date, datetime, timestamp, and time</td>
</tr>
<tr>
<td>Boolean</td>
<td>bit, and bool</td>
</tr>
<tr>
<td>Binary</td>
<td>tinyblob, mediumblob, blob, longblob, and varbinary</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The table selected for synchronization.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>writeMode</td>
<td>Select an import mode. The replace mode and insert ignore mode are supported.</td>
<td>No</td>
<td>Insert ignore</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The fields of the target table in which data is required to be written. These fields are separated by commas. For example: &quot;column&quot;: [&quot;id&quot;, &quot;name&quot;, &quot;age&quot;]. Use * if it is required to write data into all columns in sequence. For example: &quot;column&quot;: [&quot;*&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement that is run before the data synchronization task is run. Currently, you can run only one SQL statement in wizard mode, and more than one SQL statement in script mode. For example, clear old data.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement that is run after the data synchronization task is run. Currently, you can run only one SQL statement in wizard mode, and more than one SQL statement in script mode. For example, add a timestamp.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>batchSize</td>
<td>The quantity of records submitted in one operation. This parameter can greatly reduce the interactions between Data Integration and MySQL over the network, and increase the overall throughput. However, the running process of Data Integration may become out of memory (OOM) if the value is too large.</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Data source:

   Configuration item descriptions:

   ![Data Source Configuration Diagram]

   Parameters:

   - Data source: The datasource in the preceding parameter description. Enter the data source name you configured.
   - Table: The table in the preceding parameter description. Select the table for synchronization.
   - Prepared statement before import: preSQL in the preceding parameter description, namely, the SQL statement run before the data synchronization task.
   - Post-import completion statement: postSQL in the preceding parameter description, which is the SQL statement that is run after the data synchronization task is run.
2. Field mapping: The column in the preceding parameter description.

The Source Table Field on the left maps with the Target Table Field on the right. Click Add Line, and then a field is added. Hover the cursor over a line, click Delete, and then the line is deleted.

- In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.

- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Channel control

Parameters:

- Concurrent count: The maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- Number of error records: The maximum number of dirty data records.
- Task resource group: The machine on which the task runs, if the number of tasks is large. The default Resource Group is used to wait for a resource, it is recommended that you add a Custom Resource Group. For more information, see #unique_12.
Development in script mode

**Configure a job to write data into DRDS:**

```json
{
   "type": "job",
   "version": 2.0, // version number
   "steps": [
   {
      "stepType": "stream",
      "parameter": {}
   },
   {
      "stepType": "drds", // plug-in name
      "parameter": {
         "postSql": [], // SQL statement executed after the data synchronization task is executed
         "datasource": "", // Data Source
         "column": ["id"],
         "writeMode": "insert ignore ",
         "batchSize": "1024", // number of records submitted in one batch size
         "table": "test", // table name
         "postSql": [], // SQL statement executed after the data synchronization task is executed
      }
   },
   "name": "Writer",
   "category": "writer"
   ],
   "setting": {
      "errorLimit": {
         "record": "0" // Number of error records
      },
      "speed": {
         "throttle": false, // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
         "concurrent": "1", // Number of concurrency
         "dmu": 1 // Number of DMU
      }
   },
   "order": {
      "hops": [
      {
         "from": "Reader",
         "to": "Writer"
      }
      ]
   }
}
```
1.6.5.5 Configure FTP Writer

This topic describes the data types and parameters supported by FTP Writer and how to configure Writer in both wizard mode and script mode.

FTP Writer is used to write one or more files in CSV format to a remote FTP file. At the underlying implementation level, FTP Writer converts the data under the Data Integration transfer protocol to CSV files and writes these files to the remote FTP server using FTP-related network protocols. You must configure the data source before configuring the FTP Writer plug-in.

Note:
For more information, see Configure FTP data source.

What is written and saved to the FTP file is a two-dimensional table in a logic sense, for example, text information in CSV format.

FTP Writer provides the function to convert the Data Integration protocol to a FTP file. The FTP file is a non-structured data storage file. FTP Writer supports the following features:

- Only supports writing text files (BLOB, for example, video data is not supported) and schema in the text file must be a two-dimensional table.
- Supports CSV and text files with custom delimiters.
- Does not support text compression during writing.
- Supports multi-thread writing, with different subfiles written using different threads.

The following two features are not supported for the time being.

- FTP does not provide data types.
- FTP Writer writes data of String type to FTP file.

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>timeout</td>
<td>Time-out period in milliseconds of the connection to the FTP server.</td>
<td>No</td>
<td>60000 (1 minute)</td>
</tr>
<tr>
<td>path</td>
<td>The FTP file system path. The FTP Writer writes multiple files under the path directory.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>FileName</td>
<td>The file name written by FTP Writer. A random suffix is appended to the file name to form the actual file name written with each thread.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
| writeMode    | The mode in which FTP Writer clears existing data before writing data. Options include:  
  - truncate: Clear all the files prefixed by fileName in the directory before writing.  
  - append: The file is not processed before writing, and Data Integration FTP Writer writes data directly using fileName without conflict of file names.  
  - nonConflict: An error is reported if a file prefixed by fileName exists under the path directory. | Yes     | None          |
| fieldDelimiter | The delimiter used to separate the written fields.                           | Yes. A single character is used.          | None          |
| compress     | The gzip and bzip2 compression modes are supported.                          | No       | Do Compress   |
| encoding     | Encoding of the read files.                                                 | No       | UTF-8         |
| nullFormat   | Defining null (null pointer) with a standard string is not allowed in text files. Data Integration provides nullFormat to define which strings can be expressed as null.  
  For example, if you configure `nullFormat="null"`, then if the source data is null, data integration is considered a null field. | No       | None          |
<p>| dateFormat   | The format in which the data of Date type is serialized into file, for example, &quot;dateFormat&quot;: &quot;yyyyMMdd&quot;. | No       | None          |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileFormat</td>
<td>The format written by the file includes both CSV and text, and the CSV is a strict CSV format. If you want to write data that includes the column separator, it is escaped in the escape syntax of the CSV. The escape symbol is double quotes. The text format is a simple division of the data to be written using the column separator, do not escape for data to be written, including column separator.</td>
<td>No</td>
<td>text</td>
</tr>
<tr>
<td>header</td>
<td>The header used when a txt file is written, for example, 'id', 'name', 'age'.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>MarkDoneFile</td>
<td>The name of the file marked as &quot;done&quot;. After a synchronization task is completed, a MarkDoneFile is generated, based on whether the task is executed successfully is determined.</td>
<td>No</td>
<td>None</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

Configuration item descriptions:

Parameters:

- **Data source**: The datasource in the preceding parameter description. Select the FTP data source.
- **File path**: The path in the preceding parameter description.
- **Column delimiter**: The fieldDelimiter in the preceding parameter description, which defaults to a comma (,).
- **Encoding format**: The encoding in the preceding parameter description, which defaults to utf-8.
- **Null value**: The nullFormat in the preceding parameter description, which is used to define a string that represents the null value.
- **Compression format**: The compress in the preceding parameter description, which defaults to "no compression".
- **Whether to include the table header**: The **skipHeader** in the preceding parameter description, which defaults to "No".
- **Prefix conflict**: The writemode in the above parameter description defines a string that represents a null value.
2. Field mapping: The column in the preceding parameter description.

The source table field on the left and the target table field on the right are one-to-one relationships, click Add Line to add a single field and click Delete to delete the current field.

In-row mapping: You can click In-row mapping to create a mapping for the same row. Note that the data type must be consistent.

3. Channel control

Parameters:

- DMU: A unit which measures the resources consumed during data integration, including CPU, memory, and network bandwidth. One DMU represents the minimum amount of resources used for a data synchronization task.
- Concurrent job count: Maximum number of threads used to concurrently read or write data into the data storage media in a data synchronization task. In wizard mode, configure a concurrency for the specified task on the wizard page.
- The maximum number of errors means the maximum number of dirty data records.
- Task resource group: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend you add a Custom Resource Group. For more information, see Add scheduling resources.
Development in script mode

Configure synchronization jobs written to the FTP database.

```json
{
    "type": "job",
    "version": "2.0", // version number
    "steps": [
        {// The following is a reader template. You can find the corresponding reader plug-in documentations.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "ftp", // plug-in name
            "parameter": { // plug-in name
                "path": "" // File path
                "fileName": "" // File name
                "nullFormat": "null", // Null Value
                "dateFormat": "yyyy-MM-dd HH:mm:ss", // time format
                "datasource": "" // Data Source
                "writeMode": "" // Write mode
                "fieldDelimiter": "," // Separator of each column
                "encoding": "UTF-8", // Encoding format
                "fileFormat": "" // File type
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": { // Setting
        "errorLimit": { // Error Limit
            "record": "0" // Number of error records
        },
        "speed": { // Speed
            "throttle": false, // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
            "concurrent": "1", // Number of concurrent tasks
            "dmu": 1 // DMU Value
        }
    },
    "order": { // Order
        "hops": [{ // Hops
            "from": "Reader",
            "to": "Writer"
        }]
    }
}
```
1.6.5.6 Configure HBase Writer

This topic describes the data types and parameters supported by Stream Writer and how to configure Writer in script mode.

The HBase Writer plug-in provides the function to write data into HBase. At the underlying implementation level, HBase Writer connects to a remote HBase service through the HBase Java client, and writes data into HBase in put mode.

Supported features

- HBase0.94.x and HBase1.1.x versions are supported
  - If you use HBase 0.94.x, choose HBase094x as the Writer plug-in. For example:

    ```json
    "writer": {
      "hbaseVersion": "hbase094x"
    }
    ```
  
  - If you use HBase 1.1.x, choose HBase11x as the Writer plug-in. For example:

    ```json
    "writer": {
      "hbaseVersion": "hbase11x"
    }
    ```

- Multiple fields in the source end can be concatenated into a rowkey
  
  Currently, HBase Writer can concatenate multiple fields in the source end into the rowkey of an HBase table. For details, see the rowkeyColumn configuration.

- Support to versions of data written into HBase
  
  Supported timestamps (versions) for data written into HBase include:

  - Current time
  - Specified source column
  - Specified time

HBase Reader supports HBase data types and converts HBase data types as follows:

<table>
<thead>
<tr>
<th>Data integration internal types</th>
<th>Hbase data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>int, short, long</td>
</tr>
<tr>
<td>float, double</td>
<td>float, double</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
</tbody>
</table>
### Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>haveKerberos</td>
<td>If haveKerberos is True, the HBase cluster needs to be authenticated using kerberos.</td>
<td>No</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If this value is configured as true, the following five parameters related to kerberos authentication must be configured: kerberosKeytabFilePath, kerberosPrincipal, hbaseMasterKerberosPrincipal, hbaseRegionserverKerberosPrincipal, and hbaseRpcProtection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If the HBase cluster is not authenticated using kerberos, these six parameters are not required.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hbaseConfig</td>
<td>Configuration required for connecting to the HBase cluster in JSON format. The required item is hbase.zookeeper.quorum, which means the URL of HBase ZK. In addition, more HBase client configurations can be added. For example, you can configure the cache and batch of scan to optimize the interaction with servers.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>mode</td>
<td>The mode in which data is written into HBase. Currently, only the normal mode is supported. The dynamic column mode is still under development.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>Name of the HBase table to be written. The name is case sensitive.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>encoding</td>
<td>The encoding method is UTF-8 or GBK, which is used when data in string is converted to HBase byte[].</td>
<td>No</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| column    | The HBase field to be written.  
|           | - index: Specifies the index of the column that corresponds to the column of the Reader, starting from 0.  
|           | - name: Specifies the column in the HBase table, which must be in column family:column name format.  
|           | - type: Specifies the type of data to be written, which is used to convert HBase byte[]. |
|           | Yes         |
|           | N/A         |
| maxVersion| Specifies the number of versions of data to be read by HBase Reader in multi-version mode, which can only be -1 (to read all versions) or a number larger than 1. |
|           | The configuration format is as follows:  
<p>|           | None       |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>Specifies the rowkey range that the hbase reader reads.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• startRowkey: Specifies start rowkey.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• endRowkey: Specifies end rowkey.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• isBinaryRowkey: Specifies the way in which the configured startrowkey and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>endrowkey are converted to byte, the default is false. If it is true,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bytes.toBytesBinary(rowkey) is called for conversion. If it is false,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bytes.toBytes(rowkey) is called.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The configuration format is as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;range&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;startRowkey&quot;:&quot;aaa&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;endRowkey&quot;:&quot;ccc&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;isBinaryRowkey&quot;:false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The format of the configuration file is as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;column&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;index&quot;:1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;cf1:q1&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;index&quot;:2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;cf1:q2&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rowkeyColumn</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Rowkey column of the hbase to write.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• index: Specify the column index that corresponds to the Reader column,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>starting from 0. If it is a constant, index is -1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• type: Specifies the data type to be written, which is used to convert</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HBase byte[].</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• value: A configuration constant, which is usually used as the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>concatenation operator of multiple fields. HBase Writer concatenates all</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>columns of the rowkeyColumn into a rowkey in the configuration sequence</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>to write data into HBase. The rowkey cannot contain constants only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The format of the configuration file is as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;rowkeyColumn&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;index&quot;:0,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>walFlag</td>
<td>When committing data to the RegionServer in the cluster (Put/Delete operation), the HBase client writes the WAL (Write Ahead Log, which is an HLog shared by all Regions on a RegionServer). The HBase client writes data into MemStore only after it successfully writes data into WAL. In this case, the client is notified that the data is successfully committed. In case of failure to write the WAL, HBase Client is notified that the commit failed. Disable walFlag (false) to stop writing the WAL so as to improve the data writing performance.</td>
<td>No</td>
<td>false</td>
</tr>
<tr>
<td>writeBufferSize</td>
<td>Set the buffer size (in byte) of the HBase client. Use it with autoflush.</td>
<td>No</td>
<td>8 MB</td>
</tr>
</tbody>
</table>

Development in wizard mode

Currently, development in wizard mode is not supported.

Development in script mode

**Configure a job to write data from a local machine into hbase1.1.x:**

```json
{
    "type": "job",
    "version": 2.0 , // version number
    "steps": [
        { //The following is a reader template. You can find the corresponding reader plug-in documentations.
            "stepType": "stream",
            "parameter":{},
            "name": "Reader",
            "category": "reader"
        }
    ]
}
```
"stepType":"hbase", plug-in name
"parameter":{
    "mode":"normal", // mode written to hbase
    "walFlag":"false", // close (false) give up writing Wal
    "hbaseVersion": "094x", // Hbase version
    "rowkeyColumn": [] // The rowkey column of the hbase to write.
    
    "nullMode":"skip", // How do I handle null values read by "Skip"
    "column": [] // The hbase field to write.
    
    "name": "columnFamilyName1:columnName1", // field name
    "index": "0", // Index Number
    "type": "string" // data type
},

    "name": "columnFamilyName2:columnName2",
    "index": "1",
    "type": "string"
},

    "name": "columnFamilyName3:columnName3",
    "index": "2",
    "type": "string"
},

"writeMode": "api", // write mode is
"encoding": "utf-8", // encoding format
"table": ", // table name
"hbaseConfig": { // configuration information required to connect to the hbase cluster, JSON format.
    "hbase.zookeeper.quorum":"hostname",
    "hbase.rootdir":"hdfs://ip:port/database",
    "hbase.cluster.distributed":"true"
}
},

"name":"Writer",
"category":"writer"}
1.6.5.7 Configure HBase11xlsx Writer

This topic describes the data types and parameters supported by HBase11xlsx Writer and how to configure Writer in script mode.

HBase11xlsx Writer provides the function to import data in batch to an SQL table (Phoenix) in HBase. The rowkey has been encoded by Phoenix. Therefore, you need to manually convert the data when you directly use HBase APIs for data writing, which is troublesome and error-prone. This plug-in provides a method for you to import data to a single SQL table.

At the underlying implementation level, the JDBC drive of Phoenix executes the UPSERT statement to write data to HBase.

Supported functions

The writer supports importing data from an indexed table and simultaneously updating all indexed tables.

Limits

The limitations of the glaswriter plug-in are shown below.

- Only HBases of the 1.x version are supported.
- Only tables created by Phoenix are supported. Native HBase tables are not supported.
- Data with a timestamp cannot be imported.

Implementation principles

The JDBC drive of Phoenix executes the UPSERT statement to write data in batch to a table. Because an upper-layer API is used, the indexed tables can be updated simultaneously.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>plugin</td>
<td>The plug-in name, which must be hbase11xsql.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The table name to be imported. The name is case sensitive and the Phoenix tables name is generally in upper case.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>column</td>
<td>The column name. The name is case sensitive and the name of Phoenix tables is generally in upper case.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The column sequence must exactly correspond to the sequence of columns output by the reader.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The data type does not need to be entered, and the column metadata is automatically retrieved from Phoenix.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hbaseConfign</td>
<td>The address of the HBase cluster in the format of ip1,ip2,ip3. The zk is required.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Separate multiple IP addresses by commas (,).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• znode is optional and the default value is / hbase.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>batchSize</td>
<td>The maximum number of rows written in bulk.</td>
<td>No</td>
<td>256</td>
</tr>
<tr>
<td>nullMode</td>
<td>Specifies the processing mode when the column value read is null. There are currently two methods:</td>
<td>No</td>
<td>skip</td>
</tr>
<tr>
<td></td>
<td>• - skip: Skip this column. This column is not inserted. If this column of the row already exists, the column is deleted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• - empty: Insert a null value. 0 is inserted for the numeric type value and a null string is inserted for a varchar value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Development in script mode

The script configuration example is as follows.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "setting": {
            "errorLimit": {
                "record": "0"
            },
            "speed": {
                "mbps": "1",
                "concurrent": "1"
            }
        },
        "reader": {
            "plugin": "odps",
            "parameter": {
                "datasource": "",
                "table": "",
                "Column ":[
                    "partition": ""
                ]
            }
        },
        "plugin": "hbase11xsql",
        "parameter": {
            "table": "Name of the target HBase table, which is case sensitive",
            "hbaseConfig": {
                "hbase.zookeeper.quorum": "Address of the ZK server of the target HBase cluster. Ask PE for the address",
                "zookeeper.znode.parent": "znode of the ZK server of the target HBase cluster. Ask PE for the znode"
            },
            "column": ["columnName"
                ,"batchSize": 256,
                "nullMode": "skip"
            ]
        }
    }
}
```

Limits

The column sequence in the Writer must match that in the Reader. The Reader column sequence defines the sequence of columns in each row. The column sequence in the Writer defines the column sequence of the received data that is expected by the Writer. For example:

If the column sequence in the Reader is c1, c2, c3, c4, and the column sequence in the Writer is x1, x2, x3, x4, the Reader outputs column c1 to column x1 in the Writer. If the Writer column sequence is x1, x2, x4, x3, then x4 is assigned to c3, and c4 is assigned to x3.
FAQ

Q: How many concurrent settings are appropriate? Can I increase the concurrency to accelerate the import speed?

A: The default JVM stack size for the data import process is 2 GB, and the concurrency (number of channels) is realized by multiple threads. Too many threads sometimes cannot accelerate the import speed, but may result in performance deterioration due to frequent GC. A recommended concurrency (number of channels) is 5 to 10.

Q: What should the batchSize value be?

A: The default value is 256. You should set an appropriate batchSize according to the data volume in each row. Generally, the data volume at one operation is about 2 MB to 4 MB. You should divide this value by the data volume in the row and set the batchSize accordingly.

1.6.5.8 Configure HDFS Writer
This topic describes the data types and parameters supported by HDFS Writer and how to configure the Writer in Script Mode.

The HDFS Writer is used to write TextFile, ORCFile, and ParquetFile to the specified path in HDFS. The files can be associated with Hive tables. You must configure the data source before configuring the HDFS Writer plug-in. For more information, see Configure FTP data source.

How to implement HDFS Writer

The implementation process for HDFS Writer as follows:

1. Create a temporary directory that does not exist in HDFS based on the specified path.

   Naming rule: path_random

2. Write files that have been read to this temporary directory.

3. When all files are written to the temporary directory, move these files to the directory you specified. The file names should be unique.

4. Delete the temporary directory. If you are unable to connect to HDFS because of network interruptions or other reasons, delete the temporary directory and the files written to it manually.
Note:
For data synchronization, admin account, and read/write permissions for the files are required.

As shown in the preceding figure:

- Create an admin user and home directory, specify a user group and additional group, and grant the files permissions.
  
  ```bash
  useradd -m -G supergroup -g hadoop -p admin admin
  ```
  
  - **-G supergroup**: Specifies the additional group to which the user belongs.
  - **-g hadoop**: Specifies the user group to which the user belongs.
  - **-p admin admin**: Add a password to the admin user.

- View the files contents in this directory.
  
  ```bash
  hadoop fs -ls /user/hive/warehouse/hive_p_partner_native
  ```

When using Hadoop commands, the format is `hadoop fs -command`, where command represents the command.
• Copy the file part-00000 to the local file system.
  ```bash
  hadoop fs -get /user/hive/warehouse/hive_p_partner_native/part-00000
  ```
• Edit the copied file.
  ```bash
  vim part-00000
  ```
• Exit the current user.
  ```bash
  exit
  ```
• Connect to the host from the list and create an admin account for each attached host.
  ```bash
  pssh -h /home/hadoop/slave4pssh useradd -m -G supergroup -g hadoop -p admin admin
  ```
  - pssh -h /home/hadoop/slave4pssh: Connect to the host from the manifest file.
  - useradd -m -G supergroup -g hadoop -p admin admin: Create an admin account.

Functional restrictions

• HDFS Writer only supports TextFile, ORCFile, and ParquetFile formats. Content stored in the file must be a two-dimensional table in a logic sense.
• HDFS is a file system with no schema. Therefore, it does not support writing columns partially.
• Only the following Hive data types are supported:
  - numeric: TINYINT, SMALLINT, INT, BIGINT, FLOAT, and DOUBLE
  - string: STRING, VARCHAR, and CHAR
  - boolean: BOOLEAN
  - time type: date, timestamp.
• Currently, Hive data types such as Decimal, Binary, Arrays, Maps, Structs, and Union are not supported.
• For Hive partition tables, the data can only be written to one partition at a time.
• For the TextFile format, ensure delimiters in the files written to HDFS are identical to those used in the tables created in Hive, so the data written to HDFS is associated with the Hive table fields.
In the current plug-in, the Hive version is 1.1.1 and the Hadoop version is 2.7.1. Apache is compatible with JDK1.7. Data can be written normally in the testing environments of Hadoop 2.5.0, Hadoop 2.6.0, and Hive 1.2.0. For other versions, further tests are needed.

Data type conversion

Currently, HDFS Writer supports most Hive data types. Check whether the Hive type is supported.

HDFS Writer converts Hive data types as follows:

<table>
<thead>
<tr>
<th>Data Integration category</th>
<th>HDFS/Hive data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>TINYINT, SMALLINT, INT, BIGINT</td>
</tr>
<tr>
<td>double</td>
<td>FLOAT, DOUBLE</td>
</tr>
<tr>
<td>string</td>
<td>STRING, VARCHAR, CHAR</td>
</tr>
<tr>
<td>boolean</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>date</td>
<td>DATE, TIMESTAMP</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultFS</td>
<td>The namenode address in Hadoop HDFS, for example: hdfs://127.0.0.1:9000. The default resource group does not support the configuration of the advanced Hadoop parameter HA.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>fileType</td>
<td>The file type. Currently, only text, orc, and parquet are supported.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>· text: Indicates the TextFile.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· orc: Indicates the ORCFile.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>· parquet: Indicates the ParquetFile.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>path</td>
<td>The path in which the files are written to Hadoop HDFS. The HDFS Writer writes multiple files under the path based on the concurrent writing configurations. To associate a Hive table, enter the path of the Hive table stored in HDFS. For example, if the path to the data warehouse set in Hive is <code>/user/hive/warehouse/</code> and the created database test table is named hello, the Hive table path is <code>/user/hive/warehouse/test.db/hello</code>.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>FileName</td>
<td>The file name written by HDFS Writer. A random suffix is appended to the file name to form the actual file name written with each thread.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The fields of the written data. Some columns cannot be written. To associate a Hive table, you must specify all field names and table types, and specify name and type of the field name and type respectively. You can configure the column field as follows:</td>
<td>Yes.</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>&quot;column&quot;:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;userName&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;age&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;long&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>writeMode</td>
<td>The mode in which the HDFS Writer clears existing data before writing data:</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>• append: The file is not processed before writing , and the Data Integration HDFS Writer writes data directly using fileName without conflict in file names.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• nonConflict: An error is reported, if a file prefixed by fileName exists under the path directory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="Image" alt="Note:" /> Parquet files only support nonConflict mode, and does not support the Append mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fieldDelimiter</td>
<td>The field delimiter used for the fields written by HDFS Writer. Ensure the field delimiter is identical to the one used in the Hive table created . Otherwise, you are unable to locate data in the Hive table.</td>
<td>Yes.</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compress</td>
<td>The compression type of HDFS files. By default, it is left empty, which means no compression is performed. Text files support gzip and bzip2 compression types. Orc files support SNAPPY compression and requires SnappyCodec.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>encoding</td>
<td>The encoding configuration for the Write File.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default Value</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compression</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**encoding**

The encoding configuration for the Write File.

| Required | No |
| Default Value | No compression |

**compress**

The compression type of HDFS files. By default, it is left empty, which means no compression is performed.

<p>| Required | No |
| Default Value | None |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>parquetSchema</td>
<td>This parameter is required for parquet format files and is used to specify</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>the structure of the target file. This parameter takes effect only when the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fileType is parquet. The format is as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>message MessageType {</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Required, data type, column name;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters:

- **MessageType**: Any supported value.
- **Required**: Required or Optional. Optional is recommended.
- **Data Type**: Parquet files support the following data types: BOOLEAN, Int32, Int64, Int96, FLOAT, DOUBLE, BINARY (select binary if the data type is string), and fixed_len_byte_array.

**Note:**

All configuration rows and columns, including the last one must end with a semicolon.

**Example:**

```java
message m {
  optional int64 id;
  optional int64 date_id;
  optional binary datetimestring;
  optional int32 dspId;
  optional int32 advertiserId;
  optional int32 status;
  optional int64 bidding_req_num;
  optional int64 imp;
  optional int64 click_num;
}
```

**Development in Wizard Mode**

**Currently, development in Wizard Mode is not supported.**
Development in Script Mode

The script configuration example is as follows, please refer to the above parameter descriptions for details.

```json
{
    "type": "job",
    "version": 2.0, // version number
    "steps": [
        {//The following is a reader template. You can find the corresponding reader plug-in documentations.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {//stepType: "hdfs", // plug-in name
            "stepType": "hdfs", // plug-in name
            "parameter": {
                "path": "", // path information stored to hadoop HDFS
                "fileName": ",//HDFS writer file name when writing
                "compress": "", // HDFS File compression type
                "datasource": "", //Name of the data source
                "column": [
                    {"name": "col1", // field name
                        "type": "string" // Field Type
                    },
                    {"name": "col2",
                        "type": "int"
                    },
                    {"name": "col3",
                        "type": "double"
                    },
                    {"name": "col4",
                        "type": "boolean"
                    },
                    {"name": "col5",
                        "type": "date",
                    }
                ],
                "writeMode": "insert", //Write mode
                "fieldDelimiter": ",", //Delimiter of each column
                "Encoding": "UTF-8", // encoding format
                "fileType": "text" // text type
            }
        },
        {"name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": "0" //Number of error records
        },
        "speed": {
            "concurrent": "3", //Number of concurrent tasks
        }
    }
}
```
1.6.5.9 Configure MaxCompute Writer
This topic describes the data types and parameters supported by MaxCompute Writer and how to configure Writer in both wizard and script modes.

The MaxCompute Writer plug-in is designed for ETL developers to insert or update data in MaxCompute and has the capability to import business data to MaxCompute. This plug-in is suitable for TB and GB-level data transmission.

Note:
Before you start configuring the MaxCompute writer plug-in, first configure the data source. For more information, see Configure MaxCompute data source.

For more information on MaxCompute, see Introduction to MaxCompute.

At the underlying implementation level, MaxCompute Writer writes data into MaxCompute by using Tunnel based on the source project, table, partition, table field, and other configured information. For more information on common, see Tunnel Command Operations.

Supported data type

MaxCompute Writer supports the following data types in MaxCompute:

<table>
<thead>
<tr>
<th>Data</th>
<th>MaxCompute data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Bigint</td>
</tr>
<tr>
<td>Float</td>
<td>Double and decimal</td>
</tr>
<tr>
<td>String type</td>
<td>String</td>
</tr>
<tr>
<td>Date and time</td>
<td>Datetime</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
</tbody>
</table>
### Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The data table name to write data into is case-insensitive. Writing data into multiple tables is not supported.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
| partition | The data table partition information must be written. Specify the parameter until the last-level partition. For example, to write data in a three-level partition table, you must configure to the last-level partition, for example, pt=20150101, type=1, biz=2.  
  - This parameter is not required for non-partition tables, this results in the data directly imported to the target table.  
  - MaxCompute Writer does not support writing data by routing. For partition tables, always ensure data is written through to a last-level partition. | Required if the table is a partition table. This can be left empty in non-partition tables. | None |
| column    | A list of fields that need to be imported, which can be configured as "column": ["*"] when all fields are imported ".: ["*"]. When you need to insert a partial MaxCompute column, enter a partial column, for example, "column": ["id", "name"].  
  - MaxCompute writer supports Column Filtering, column switching, for example, there are three fields in a table, A, B, and C. You can configure to "column": ["c","b"] by synchronizing only the C and B fields. During the import process, field A is automatically empty, and set to null.  
  - Column must contain the specified column set to be synchronized and it cannot be blank. | Yes      | None          |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>truncate</td>
<td>&quot;truncate&quot;: &quot;true&quot; is configured to ensure the idempotent of write operations. When a reattempt is made after a failed write attempt, MaxCompute Writer cleans up this data and imports new data. This ensures the data is consistent after each rerun. The option truncate is not an atomic operation. SQL cannot be atomic because MaxCompute SQL is used for data cleansing. Therefore, when multiple tasks clean up a Table/Partition at the same time, the concurrency and timing problem may occur. So proceed with caution. To avoid this problem, we recommend that you do not operate on one partition with multiple job DDLs at the same time, or that you create partitions before starting multiple concurrent jobs.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

Configuration item descriptions:

Parameters:

- Data source: The datasource in the preceding parameter description. Enter the configured data source name.
- Table: The table in the preceding parameter description. Select the table for synchronization.
- Partition information: If all columns are specified, you can configure them in column, for example, "column": ["""]. Partition supports configuration methods that configure multiple partitions and wildcard characters.
  
  - "partition": "pt=20140501/ds=*" Represents all partitions in DS.
  - "partition": "pt=top?" In? indicates whether the character in front of it exists. This configuration specifies the two partitions with pt=top and pt=to.

You can enter the partition columns for synchronization, such as partition columns with pt. For example: Assume the value of each MaxCompute partition is pt=${bdp.system.bizdate}, add the partition name pt to a field in the source table, ignore the unrecognized mark if any, and proceed to the next
step. To synchronize all partitions, configure the partition value to `pt=${*}`. To synchronize a specific partition, select a time value for the partition.

- Clearance rules:
  - Clear Existing Data Before Import: All data in the table or partition is cleared before import, which is equivalent to insert overwrite.
  - Keep Existing Data Before Writing: No data is cleared before data import. New data is always appended with each run, which is equivalent to "Insert into".

- Compression: Default selection is not compressed.
- Whether the empty string is null: The default setting is yes.

2. The field mapping is the column in the above parameter description.

The source table field on the left and the target table field on the right have a one-to-one relationships, click Add row to Add a Single field and click Delete to delete the current field.

- In-row mapping: You can click In-row Mapping to create a mapping for the same row. Note that the data type must be consistent.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Control the tunnel

Parameters:

- Concurrent job count: The maximum number of threads used to concurrently read/write data into the data storage media in a data synchronization task.
In Wizard Mode, configure a concurrency for the specified task on the Wizard page.

- The maximum number of errors indicates the maximum number of dirty data records.
- Task resource group: The machine on which the task runs. If there is a large number of tasks, the default Resource Group is used to wait for a resource, it is recommended that you add a Custom Resource Group. For more information, see Add scheduling resources.

Development in Script Mode

The following example is a script configuration. Please refer to the preceding parameter descriptions for details.

```json
{
    "type": "job",
    "version": 2.0, // version number
    "steps": [
        {//You can locate the corresponding writer plug-in documentation among the following documentations.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader ",
            "category": "reader"
        },
        {// plug-in name
            "stepType": "odps", // plug-in name
            "parameter": {
                "partition": ",//Shard information
                "truncate": true, // write Rule
                "compress": false, // do you Want to compress?
                "datasource": "odps_first",//The data source name.
                "column": [ // column name
                    "*
                ],
                "emptyAsNull": false, if the empty string is null?
                "table": ",// table name
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "Setting ":{
        "errorLimit": {
            "record": 0 //Maximum number of error records
        },
        "speed": {
            "throttle": false, // do you want to limit the flow?
            "concurrent": 1, //Number of concurrent tasks
            "dmu": 1 // DMU Value
        }
    },
    "order":{
        "hops":[
            
```
Questions about Column Filtering

MaxCompute does not support column filtering, reordering, and null filling, but MaxCompute Writer does. For example, a list of fields that need to be imported can be configured as "column": ["*"], when all fields are imported. "column": ["C", "B"], indicates the first and second columns of reader will be imported into the C and B fields of MaxCompute. The newly inserted a field in the MaxCompute table is set to null.

Column configuration error handling

To ensure data is written in a reliable manner, data loss from redundant columns must be prevented to avoid data quality failure. When redundant columns are written, MaxCompute Writer produces an error. For example, the MaxCompute Writer will generate an error when the MaxCompute table has fields A, B, and C, but the MaxCompute Writer writes more than three fields.

Partition configuration

MaxCompute Writer only provides the write through to a last-level partition function, and does not support partition routing of writing based on a specific field. For a table that has three levels of partition, you must specify writing data to a level-3 partition. For example, write data to the level-3 partition of a table. You can configure it to pt=20150101, type=1, biz=2, but not pt=20150101, type=1 or pt=20150101.

Task rerun and failover

In MaxCompute Writer, "truncate": true is configured to ensure the idempotent of write operations. When a reattempt is made after a failed write attempt, MaxCompute Writer cleans up this data and imports new data. This ensures data is consistent after each rerun. If the task is interrupted by any exceptions during the run process, the data atomicity cannot be guaranteed, nor will data be rolled back.
or rerun automatically. It is required that you use this idempotent to rerun the task to ensure data integrity.

Note:
Setting "truncate" to "true" cleans up all data of the specified partition or table, so proceed with caution.

1.6.5.10 Configure Memcache (OCS) Writer
This topic describes the data types and parameters supported by Memcache (OCS) Writer and how to configure Writer in script mode.

ApsaraDB for Memcache (formerly known as OCS) is a seamless scalable distributed memory database service with high performance and reliability. Based on the Apsara distributed system and high performance storage, ApsaraDB for Memcache provides a complete set of solutions for active/standby hot standby, disaster recovery, business monitoring, data migration, and other scenarios.

ApsaraDB for Memcache supports out-of-the-box deployment mode, and alleviates database load for dynamic web applications using the cache service, thus accelerating the overall website response.

Similar to local Memcache databases, ApsaraDB for Memcache is compatible with the Memcached protocol. You can use it directly in the operating environment. The difference is that the hardware and data of ApsaraDB for Memcache are deployed in the cloud, providing complete infrastructure, network security, and system maintenance services. All these services are billed on a Pay-As-You-Go basis.

Memcache Writer writes data into Memcache channels based on the Memcached protocol.

Currently, Memcache Writer supports only one write mode. Data types written in different modes are converted differently:

- text: Memcache Writer serializes source data to the String type, and uses your fieldDelimiter as the delimiter.
- Binary: Data type is not supported.
Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>writeMode</td>
<td>Memcache Writer writes data in the following modes:</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>- set: Stores the data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- add: Stores the data only when the key does not exist (currently is not supported).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- replace: Stores the data only when the key exists (currently is not supported).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- append: Stores data after the existing key and ignores exptime (currently is not supported).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- prepend: Stores data before the existing key and ignores exptime (currently is not supported).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
| writeFormat     | Currently, Memcache Writer supports writing data in only TEXT format  

TEXT: Serializes the source data to the text format with the first field being the key written into Memcache, and all subsequent fields to the String type. Use the specified fieldDelimiter as the delimiter to concatenate the text data into a complete string and write it into Memcache.  
  For example, the source data is:

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>23</td>
<td>&quot;AMC&quot;</td>
<td>100</td>
</tr>
</tbody>
</table>

If fieldDelimiter is specified as ",", the data format written into Memcache is:

<table>
<thead>
<tr>
<th>KEY (OCS)</th>
<th>VALUE (OCS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>CDP&quot;100</td>
</tr>
</tbody>
</table>

| ExpireTime | The Memcache invalidation time. Currently, MemCache supports two types of invalidation time.  

- Unix time (expressed in number of seconds since January 1, 1970) indicates the data is invalid at a certain time point in the future.  
- The relative time (in seconds) starting from the current time point, which indicates the time length from the current time before data is invalid.  

Note:  
If the invalidation time is greater than 60*60*24*30 (30 days), the server identifies the invalidation time as the Unix time. | No | 0.0 permanently valid |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>batchSize</td>
<td>The quantity of records submitted in one operation. Setting this parameter can greatly reduce interactions between CDP and Memcache over the network, and increase the overall throughput. However, an excessively large value may cause the CDP running processes to become Out of Memory (OOM). (The current Memcache version does not support writing in batches.)</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>

Development in Wizard Mode

**Currently, development in Wizard Mode is not supported.**

Development in Script Mode

**Use the data generated from memory and imported into Memcache.**

```json
{
  "type": "job",
  "version": "2.0",
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "Oss",
      "parameter": {
        "Writeformat": "text", // memcache writer writes data
        "expireTime": 1000, // memcache value cache failure time
        "indexes": 0, // Data Source
        "writeMode": "insert", //Write mode
        "batchSize": "1000", // number of records submitted in one batch size
      },
      "name": "Writer",
      "category": "writer"
    }
  ],
  "Setting": {
    "errorLimit": {
      "record": 0 //Number of error records
    },
    "speed": {
      "throttle": false, //False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
      "concurrent": "1", //Number of concurrent tasks
    }
  }
}
```
1.6.5.11 Configure MongoDB Writer

This topic describes the data types and parameters supported by MongoDB Writer and how to configure Writer in Script Mode.

The MongoDB Writer plug-in uses MongoClient, the Java client of MongoDB, to write data into MongoDB. The latest version of Mongo has reduced the granularity of DB locks from the DB level to the document level, with powerful indexing capabilities of MongoDB. Data sources are basically able to meet the requirements of writing data to MongoDB. The data update requirements can also be implemented by configuring the business primary key.

Note:

- Before you start configuring the MongoDB Writer plug-in, configure the data source first. For more information, see Configure MongoDB data source.
- If you are using ApsaraDB for MongoDB, a root account is provided by default.
- To ensure security, Data Integration only supports using the relevant account of MongoDB for connection. Avoid using the root account as an access account when adding and using the MongoDB data source.

MongoDB Writer acquires the protocol data generated by Reader through the Data Integration framework, and converts data types supported by Data Integration to those supported by MongoDB separately. Data integration does not support array types, but MongoDB does support array type. The index of the array type is strong.

To use the MongoDB array type, you must convert the string to the array in MongoDB by using special parameter configurations before writing data into MongoDB.
Type conversion list

MongoDB Writer supports most data types in MongoDB. Check whether your data type is supported before using it.

MongoDB Writer converts the MongoDB data types as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>MongoDB data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>INT and Long</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
</tr>
<tr>
<td>String type</td>
<td>String and array</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Binary</td>
<td>Bytes</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the added data source name. Adding data source is supported in Script Mode.</td>
</tr>
<tr>
<td>Collection name</td>
<td>The collection name of MongoDB.</td>
</tr>
<tr>
<td>column</td>
<td>An array of multiple column names of a document in MongoDB.\n</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Writemode | The parameter that specifies whether to overwrite data during transmission.  
- isReplace: If this parameter is set to True, the data of the same replaceKey is overwritten. If it is set to False, the data is not overwritten.  
- replaceKey: This parameter specifies the business primary key for each record entry and is used to overwrite data (ReplaceKey must be unique and is generally the primary key in Mongo). | No | None |
| preSql | You can use "preSql":{"type":"remove"} to remove the collection. | No | None |

Development in Wizard Mode

**Currently, development in Wizard Mode is unavailable.**

Development in script mode

To configure data synchronization jobs written to MongoDB, please refer to the above parameter descriptions for details.

```
{
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "mongodb",
            "parameter": {
                "date": "",
                "column": [
                    {
                        "name": "name",
                        "type": "string"
                    },
                    {
                        "name": "age",
                        "type": "int"
                    },
                    {
                        "name": "id",
                        "type": "long"
                    }
                ]
            }
        }
    ]
}
```
"name": "wealth",
"type": "double"
},
{
"name": "hobby",
"type": "array",
"splitter": " "
},
{
"name": "valid",
"type": "boolean"
},
{
"name": "date_of_join",
"format": "yyyy-MM-dd HH:mm:ss",
"type": "date"
}
],
"writeMode": {
"isReplace": "true",
"replaceKey": "id"
"collectionName": "datax_test"
},
"name": "Writer",
"category": "writer"
],
"setting": {
"errorLimit": {
"record": "@"
},
"speed": {
"jvm throttle": true,    "concurrent": 1,
"mbp"
},
"order": {
"hops": [
{
"from": "Reader",
"to": "Writer"
}
]
}
}

1.6.5.12 Configure MySQL Writer
This topic describes the data types and parameters supported by MySQL Writer and how to configure the Writer in both Wizard and Script mode.

The MySQL Writer plug-in can write data into a target table of a MySQL database. At the underlying implementation level, MySQL Reader connects to a remote MySQL database through the JDBC, and runs the insert into... or replace into... SQL statement to write data into MySQL. Data is written into the database in batches within MySQL, and the database must use InnoDB engine.
You must configure the data source before configuring the MySQL Writer plug-in. For more information, see Configure MySQL data source.

MySQL Writer is designed for ETL developers to import data from data warehouses to MySQL. MySQL Writer can also be used as a data migration tool by DBA and other users. MySQL Writer acquires the protocol data generated by the Reader based on writeMode through the Data Synchronization framework.

Note:
The entire task requires at least the insert/replace into... permission. Whether other permissions are required depends on the statements specified in the PreSQL and PostSQL when you configure the task.

Type conversion list

Similar to MySQL Reader, MySQL Writer currently supports most data types in MySQL. Check whether your data type is supported.

MySQL Writer converts the MySQL data types as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>MySQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>int, tinyint, smallint, mediumint, int, bigint, and year</td>
</tr>
<tr>
<td>Floating point</td>
<td>float, double, and decimal</td>
</tr>
<tr>
<td>String</td>
<td>varchar, char, tinytext, text, mediumtext, and longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>date, datetime, timestamp, and time</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Binary</td>
<td>tinyblob, mediumblob, blob, longblob, and varbinary</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name entered here must be the same as the added data source. You can add a data source in script mode.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>table</td>
<td>The table selected for synchronization.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Require</td>
<td>Default value</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>writeMode</td>
<td>Selects an import mode. The insert/replace mode is supported.</td>
<td>No</td>
<td>insert</td>
</tr>
<tr>
<td></td>
<td>• replace into... If the primary key does not conflict with the unique index, the system performs insert into. When a conflict exists, all fields in the original line are replaced with the fields in the new line.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• insert into... If the primary key conflicts with the unique index, data cannot be written into the conflicting lines and is classified as dirty data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• INSERT INTO table (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE...; If the primary key does not conflict with the unique index, the system performs the same action as insert into. When a conflict exists, the specified field in the original line is replaced with the field in the new line.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>column</td>
<td>The target table fields in which data is required to be written. These fields are separated by commas (,). For example: &quot;column&quot;: [&quot;id&quot;, &quot;name&quot;, &quot;age&quot;]. Use * if it is required to write data into all columns in sequence. For example, &quot;column&quot;: [&quot;*&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement that runs before running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and more than one SQL statement in Script Mode. For example: clear old data.</td>
<td>No</td>
<td>None</td>
</tr>
</tbody>
</table>

**Note:**
If multiple SQL statements exist, the transaction is not supported.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>postSql</td>
<td>The SQL statement that runs after running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and more than one SQL statement in Script Mode. For example: add a timestamp.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Note: If multiple SQL statements exist, the transaction is not supported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in a single operation. Setting this parameter can greatly reduce interaction between Data Synchronization and MySQL, and increase the overall throughput. However, an excessively large value may cause the running process of Data Synchronization to become Out of Memory (OOM).</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

Configuration item descriptions:

Parameters:

- Data source: The datasource in the parameter description section. Enter the data source name you configured.
- Table: The table in the preceding parameter description. Select the table for synchronization.
- Prepared statement before import: The preSQL in the preceding parameter description, namely, the SQL statement that runs before running the data synchronization task.
- Post-import completion statement: The postSQL in the preceding parameter description, which is the SQL statement that runs after running the data synchronization task.
- Primary key conflict: The writeMode in the preceding parameter description. You can select the import mode.
2. Field mapping: The column in the parameter description section.

The Source Table Field on the left maps with the Target Table Field on the right. Click Add Line to add a field. To delete the line, move the cursor over a line, and click Delete.

- In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Channel control

Parameters:

- DMU: A unit that measures the resources consumed during data integration, including CPU, memory, and network bandwidth. One DMU represents the minimum amount of resources used for a data synchronization task.
- Concurrent job count: The maximum number of threads used to concurrently read/write data into the data storage media in a data synchronization task. Go to the Wizard page under Wizard mode to configure a concurrency for the specified task.
- The maximum number of errors indicates the maximum number of dirty data records.
- Task resource group: The machine on which the task runs, if the number of tasks is large. The default Resource Group is used to wait for a resource, it is recommended that you add a Custom Resource Group. For more information, see #unique_12.
Development in Script Mode

The following is an example of Script configuration. For relevant parameters, see Parameter Description.

```json
{
  "type": "job",
  "version": 2.0, // version number
  "steps": { // below is the template for reader, you can find the appropriate read plug-in documentation.
    "stepType": "stream",
    "parameter": {},
    "name": "Reader",
    "category": "reader"
  },
  { // stepType: "mysql" // plug-in name
    "stepType": "mysql", // plug-in name
    "parameter": {
      "postSql": [], // Post-import preparation statement
      "datasource": "", // Data Source
      "column": [ // column name
        "id",
        "value"
      ],
      "writeMode": "insert", // Write mode
      "batchSize": "1024", // number of records submitted in one batch size
      "table": "", // table name
      "preSql": [] // Pre-import preparation statement
    },
    "name": "Reader",
    "category": "writer"
  }
},

"setting": {
  "errorLimit": { // Number of error records
    "record": "0"
  },
  "speed": {
    "throttle": false, // do you want to limit the flow?
    "concurrent": "1", // Number of concurrency
    "dmu": 1 // DMU Value
  }
},

"order": {
  "hops": [
    {
      "name": "Reader",
      "to": "Writer"
    }
  ]
}
```
1.6.5.13 Configuring Oracle Writer
This topic describes the data types and parameters supported by Oracle Writer and how to configure Writer in both Wizard and Script mode.

The Oracle Writer plug-in provides the capability to write data into the target tables of the primary Oracle database. At the underlying implementation level, Oracle Writer connects to a remote Oracle database through JDBC, and runs the `insert` into... SQL statement to write data into the Oracle database.

Note:
You must configure the data source before configuring the Oracle Writer plug-in. For more information, see Configure Oracle data source.

Oracle Writer is designed for Extract, transform, load (ETL) developers to import data from data warehouses to Oracle. Oracle Writer can also be used as a data migration tool by Database Administrator (DBA) and other users.

Oracle Writer uses the data synchronization framework to obtain protocol data generated by the Oracle Reader. Then it connects to a remote Oracle database through JDBC, and runs the `insert` into... The SQL statement to write data into Oracle.

Type conversion list

Oracle Writer currently supports most Oracle data types. Check whether your data type is supported.

Oracle Writer converts the data types in Oracle as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>Oracle data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>NUMBER, RAWID, INTEGER, INT, and SMALLINT</td>
</tr>
<tr>
<td>Floating point</td>
<td>NUMERIC, DECIMAL, FLOAT, DOUBLE, PRECISION, and REAL</td>
</tr>
<tr>
<td>Type classification</td>
<td>Oracle data type</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>String</td>
<td>LONG, CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, CHARACTER, CHARACTER VARYING, CHAR VARYING, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING, and NCHAR VARYING</td>
</tr>
<tr>
<td>TIMESTAMP and DATE</td>
<td>Timestamp and date</td>
</tr>
<tr>
<td>Boolean</td>
<td>BIT and BOOL</td>
</tr>
<tr>
<td>Binary</td>
<td>BLOB, BFILE, RAW, and LONG RAW</td>
</tr>
</tbody>
</table>

**Parameter description**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name must be identical to the added data source name. Adding data source is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The target table name. If the table schema information is inconsistent with the user name in the preceding configuration, enter the table information in schematable format.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>column</td>
<td>The required target table fields into which data is written, where each field is separated by commas (,). For example: &quot;column&quot;: [&quot;id&quot;,&quot;name&quot;,&quot;age&quot;]. Use asterisks (<em>) to write data into all columns in sequence. For example: &quot;column&quot;: [&quot;</em>&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement that runs before running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statement in Script Mode. For example: Clear old data.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement that runs after running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statement in Script Mode. For example: Add a timestamp.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in a single operation. Setting this parameter can greatly reduce interactions between CDP and Oracle over the network, and increase the overall throughput. However, an excessively large value may cause the running process of CDP to become Out of Memory (OOM).</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Choose source

The following are configuration item descriptions:

Parameters:

- **Data source**: The datasource from the preceding parameter description. Enter the configured data source name.
- **Table**: The table in the preceding parameter description. Select the table for synchronization.
- **Prepared statement before import**: The preSQL parameter in the preceding parameter description. The SQL statement that is run before running the data synchronization task.
- **Post-import completion statement**: postSQL in the preceding parameter description, which is the SQL statement that runs after running the data synchronization task.
- **Primary key conflict**: writeMode in the preceding parameter description. You can select the expected import mode.
2. Field mapping: The column in the preceding parameter description.

The Source Table Field on the left maps with the Target Table Field on the right. To add a field, click Add Line. To delete the line, move the cursor over a line and click Delete.

- In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Channel control

Parameters:

- Concurrent job count: The maximum number of threads used to concurrently read/write data into the data storage media in a data synchronization task. In Wizard Mode, configure a concurrency for the specified task on wizard page.
- The maximum number of errors indicates the maximum number of dirty data records.
- Task resource group: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend you add a Custom Resource Group. For more information, see #unique_12.

Development in script mode

Configure a job to write data into Oracle:

```json
{
  "type": "job",
  "version": 2.0, // version number
  "steps": [
    //The following is a reader template. You can find the corresponding reader plug-in documentations.
```
"stepType": "stream",
"parameter": {},
"name": "Reader",
"category": "reader"
},
{
"stepType": "oracle", // plug-in name
"parameter": {
"postSql": [], // SQL statement executed after the
data synchronization task is executed
"datasource": "",
"session": [], // database connection session
parameters
"column": [[// Field
    "id",
    "name"
],
"encoding": "UTF-8", // encoding format
"batchSize": "1024", // number of records submitted in
one batch size
"table": ", // table name
"postSql": [] // SQL statement executed after the data
synchronization task is executed
},
"name": "Writer",
"category": "writer"
}
],
"setting":{
"errorLimit":{
"record": "0"/Number of error records
},
"speed":{
"throttle":false, //False indicates that the traffic is
not throttled and the following throttling speed is invalid. True
indicates that the traffic is throttled.
"concurrent": "1", //Number of concurrency
"dmu": 1 //DMU Value
}
},
"order":{
"hops":[
{
"from": "Reader",
"to": "Writer"
}
]
}
}

**1.6.5.14 Configure OSS Writer**

This topic describes the data types and parameters supported by OSS Writer and how to configure Writer in both Wizard and Script mode.

**OSS Writer** allows you to write one or more table files in formats similar to CSV into OSS.

---

**Note:**
You must configure the data source before configuring the OSS Writer plug-in. For more information, see Configure OSS data source.

What is written and saved to the OSS file is a two-dimensional table in a logic sense. For example, the text information can be written in CSV format.

- For more information about OSS products, see OSS Product Overview.

OSS Writer allows you to convert the data synchronization protocol to a text file in OSS, which is a non-structured data storage. Currently, OSS Writer supports the following features:

- Only writing text files is supported. The schema in the text file must be a two-dimensional table.
- File formats similar to CSV with custom delimiters is supported.
- Multi-thread writing with different subfiles written using different threads is supported.
- File rollover is supported. A file exceeding a specific size value must be switched. A file that contains lines that exceed a specific number of lines must be switched.

Currently, OSS Writer does not support the following features:

- Concurrent writing for a single file is not supported.
- OSS does not provide data types, but OSS Writer writes String type data to OSS.

OSS does not provide data types, which are defined by DataX OSS Writer.

<table>
<thead>
<tr>
<th>Type classification</th>
<th>OSS data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Long</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name entered here must be the same name as the added data source. You can add a data source in Script Mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>Object</td>
<td>The file name written by OSS Writer. The object enables the simulation of directories with file names in OSS. For example, if the data is synchronized into the OSS bucket is the test folder of test118, you only need to specify test for the object, and you do not have to specify the bucket name. The file name synchronized to the OSS end must be identical with the source end. If the value &quot;object&quot;: &quot;test/DI&quot; is used to specify the object, the object written in OSS starts with test/DI . In this command, test is the folder, DI is the file name prefix (the suffix is a random string), and the forward slash (/) is the delimiter of the simulated OSS directory.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
| writeMode | The write mode in which the OSS Writer clears existing data before writing data.  
- truncate: Clears all objects with the specified Object prefix before writing. For example, all objects with the prefix abc will be cleared, if the specified object prefix is "object": "abc"  
- append: This parameter setting will not run any processes before writing data. Data Integration OSS Writer writes data directly with the object name, and appends a random UUID suffix name to ensure there are no conflicts in the file names. For example, if the object name you specified is Data Integration, the name entered is DI_xxxxxx_xxxx_xxxx.  
- nonConflict: This parameter will report an error, if an object with the specified prefix exists in the specified path. For example, if the specified prefix is "object": "abc", an error is reported when an object starts with abc123. | Yes | None |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileFormat</td>
<td>The written file format, includes CSV and text. If the written data in CSV format contains column delimiters, the column delimiters are escaped into double quotation marks (&quot;), by the CSV escape syntax. For text format, the data written is separated by column delimiters without being escaped.</td>
<td>No</td>
<td>Text</td>
</tr>
<tr>
<td>fieldDelimiter</td>
<td>The delimiter for separating read fields.</td>
<td>No</td>
<td>,</td>
</tr>
<tr>
<td>encoding</td>
<td>Encodes written files.</td>
<td>No</td>
<td>UTF-8</td>
</tr>
<tr>
<td>nullFormat</td>
<td>You cannot define null (null pointer) with a standard string in text files. The Data Synchronization system provides nullFormat to define strings that can be expressed as null. For example, if nullFormat=&quot;null&quot; is configured, and the source data is null, the Data Synchronization system will classify it as a null field.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>header</td>
<td>The header used when a file is written in OSS. For example: ['id', 'name', 'age'].</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>maxFileSize</td>
<td>By default, the maximum size of an object file written in OSS is 10,000 x 10 MB. This parameter is similar to log rotation based on the log size in log4j log printing. Each part of a multipart upload in OSS is 10 MB in size, which is the minimum file granularity for log rotation. For maxFileSize with a size smaller than 10 MB are classified as 10 MB, and the maximum number of parts supported for each OSS InitiateMultipartUploadRequest is 10,000. When rotation occurs, the object naming rule is the original object prefix + a random UUID + a suffix, such as _1, _2, _3.</td>
<td>No</td>
<td>100,000 MB</td>
</tr>
</tbody>
</table>
Development in Wizard Mode

1. Choose source

The following is a configuration item descriptions:

Parameters:

- **Data source**: The datasource from the parameter description section. Enter the configured data source name.
- **Object prefix**: The object from the parameter description section. Enter a path to the OSS folder without the bucket name.
- **Column delimiter**: The fieldDelimiter in the preceding parameter description section. By default, the delimiter are commas (,).
- **Encoding format**: The encoding in the preceding parameter description section. By default, the encoding format is UTF-8.
- **null value**: The nullFormat in the preceding parameter description section. Defines a string that represents the null value.
2. Field mapping: The column in the preceding parameter description section.

The source table field on the left and the target table field on the right have a one-to-one relationship. To add a single filed, click Add Row. To delete the current field, click Delete.

In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.

3. Channel control

Parameters:

- DMU: A unit that measures resources consumed during data integration, including CPU, memory, and network bandwidth. One DMU represents the minimum amount of resources used for a data synchronization task.
- Concurrent job count: The maximum number of threads used to concurrently read/write data into the data storage media in a data synchronization task. You can configure a concurrency for the specified task under Wizard Mode.
- The maximum number of errors, which means the maximum number of dirty data records.
- Task resource group: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend that you add a Custom Resource Group. For more information, see #unique_12.
Development in Script Mode

The following is an example of script configuration. For details about the parameters, see the preceding Parameter Description section.

```json
{
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "oss", // plug-in name
            "parameter": {
                "nullFormat": "", // The data synchronization system provides a nullformat to define which strings can be expressed as null.
                "dateFormat": "", // Date Format
                "datasource": "", // Data Source
                "writeMode": "", // Write mode
                "encoding": "UTF-8", // encoding format
                "fieldDelimiter": ",", //Separator
                "fileFormat": "", //File type
                "object": "// object prefix"
            },
            "name": "Writer ",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": "0" //Number of error records
        },
        "speed": {
            "throttle": false, //False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
            "concurrent": "1", //Number of concurrent tasks
            "dmu": 1 //DMU Value
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader ",
                "to": "Writer"
            }
        ]
    }
}
```
1.6.5.15 Configure PostgreSQL Writer

This topic describes the data types and parameters supported by PostgreSQL Writer and how to configure Writer in both Wizard Mode and Script Mode.

The PostgreSQL Writer plug-in reads data from PostgreSQL. At the underlying implementation level, PostgreSQL Writer connects to a remote PostgreSQL database through Java Database Connectivity (JDBC) and runs corresponding SQL statements to select data from the PostgreSQL database. On the public cloud, Relational Database Service (RDS) provides a PostgreSQL storage engine.

Note:
Configure the data source before configuring a PostgreSQL Writer plug-in. For details, see Configure PostgreSQL data source.

In short, PostgreSQL Writer connects to a remote PostgreSQL database through a JDBC connector, and generates SELECT SQL query statements based on your configurations, and then sends the statements to the remote PostgreSQL database. The PostgreSQLWriter then assembles returned results of the executed SQL statement into abstract datasets through the custom CDP data types, and passes the datasets to the downstream writer.

- PostgreSQL Writer concatenates the configured table, column, and WHERE information into SQL statements and sends them to the PostgreSQL database.
- PostgreSQL directly sends the configured querySQL information to the PostgreSQL database.

Type conversion list

PostgreSQL Writer supports most PostgreSQL data types. Check whether the data type you are using is supported.

PostgreSQL Writer converts PostgreSQL data types as follows:

<table>
<thead>
<tr>
<th>Data integration internal types</th>
<th>PostgreSQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>Bigint, Bigserial, Integer, Smallint, and Serial</td>
</tr>
<tr>
<td>Double</td>
<td>Double Precision, Money, Numeric, and Real</td>
</tr>
</tbody>
</table>
Data integration internal types | PostgreSQL data type
--- | ---
String | Varchar, Char, Text, Bit, and Inet
Date | Date, Time, and Timestamp
Boolean | Bool
Bytes | Bytea

Note:
- Only the preceding field types are supported.
- To convert data types including "money", "inet", and "bit", you need to use syntaxes, such as "a_inet::varchar".

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. Adding data source is supported in Script Mode, the data source name must be the same as the added data source.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The selected table name that requires synchronization.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>writeMode</td>
<td>The specified import mode, which allows data insertion. insert: If the primary key conflicts with the unique index, Data Integration determines the data as dirty data, but retains the original data.</td>
<td>No</td>
<td>insert</td>
</tr>
<tr>
<td>column</td>
<td>The target table fields into which data is required to be written. These fields are separated by commas (,). For example: &quot;column&quot;:[&quot;id&quot;,&quot;name &quot;,&quot;age&quot;]. To write all columns subsequently, use the asterisk (<em>) for representation. For example: &quot;column&quot;:[&quot;</em>&quot;]</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSQL</td>
<td>The SQL statement that runs before running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statement in Script Mode. For example: clear old data.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>postSQL</td>
<td>The SQL statement that runs after running the data synchronization task d. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statements in Script Mode. For example: add a timestamp.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in an operation. This parameter can greatly reduce interactions between Data Integration and PostgreSQL over the network, and increase the overall throughput. However, an excessively large value may cause the running process of Data Integration to become Out of memory (OOM).</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>
Development in Wizard Mode

1. Choose source

The following is configuration item descriptions:

Parameters:

- Data source: The datasource in the preceding parameter description section. Enter the configured data source name.
- Table: The table in the preceding parameter description section. Select the table for synchronization.
- Before import: The preSQL in the preceding parameter description section, namely, the SQL statement that runs before running the data synchronization task.
- After import: The postSQL in the preceding parameter description section, which is the SQL statement that runs after running the data synchronization task.
2. Field mapping: The column in the preceding parameter description section.

The Source Table Field on the left maps with the Target Table Field on the right. To add a field, click Add Line. To delete a line, move the cursor over a line, and click Delete.

- In-row mapping: To create a mapping for the same row, click Enable Same-Line Mapping. Note the mapped data type must be consistent.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Control the tunnel

Parameters:

- Concurrent job count: The maximum number of threads used to concurrently read/write data into the data storage media in a data synchronization task. You can configure a concurrency for the specified task under Wizard Mode.
- The maximum number of errors indicates the maximum number of dirty data records.
- Task resource group: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend that you add a Custom Resource Group. For more information, see #unique_12.
The following is a script configuration sample. For details about parameters, see Parameter Description.

```json
{
    "type": "job",
    "version": 2.0 , // version number
    "steps": [] // below is the template for reader, you can find the appropriate read plug-in documentation.
    {
        "stepType": "stream",
        "Parameter ":{},
        "name": "Reader",
        "category": "reader"
    },
    {
        "stepType": "postgresql",/plug-in name
        "parameter": {
            "postSQL": [], // SQL statement that was first executed after the data synchronization task was executed
            "datasource": "// Data Source 
            "col1",
            "col2",
            ],
            "table": ", // table name
            "postSQL": [], // SQL statement that was first executed after the data synchronization task was executed
        },
        "name": "Reader",
        "category": "writer"
    }
},
"setting":{
    "errorLimit": {
        "record": "0"//Number of error records
    },
    "speed": {
        "throttle":false, //False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
        "concurrent": "1", //Number of concurrent tasks
        "dmu": 1 // DMU Value
    }
},
"order":{
    "hops":[
        {"name": "Reader",
        "To": "Writer"
    }
]}
}
```

1.6.5.16 Configure Redis Writer

This topic describes how to configure a Redis Writer. The Redis Writer is a Redis writing plug-in based on the Data Integration framework. It can import data from a
data warehouse or other data source to a Redis instance. The Redis Writer interacts with the Redis Server through Jedis, which is a preferred Java client development kit provided by Redis that nearly has all Redis features.

Remote Dictionary Server (Redis) is a high-performance log-based key-value storage system that supports both persistent or memory-based network storage. Redis can be used as a database, high-speed cache, and message queue (MQ) proxy. Redis supports different types of storage values, including string, list, set, zset (sorted set), and hash. For more information about Redis, see redis.io.

Note:

- For more information on how to configure the data source before configuring a Redis Writer plug-in, see Configure Redis data source.
- If the value are lists when writing data to a Redis instance through the Redis Writer, the rerun synchronization task result is not idempotent. If the value type is list, you must clear the related data on Redis when rerunning the synchronization task.

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The data source name must be the same as the added data source. Script Mode supports adding data source.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>keyIndexes</td>
<td>The keyIndexes indicates which columns of the source table are used as key (starts with 0 in the first column). If the key is the group of the first and second columns, the value of keyIndexes is [0,1].</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>

Note:
When keyIndexes are configured, the Redis Writer will list the remaining columns as value. You can specify the column on the Reader plug-in side for column filtering to synchronize a few columns in the source table as key and a few columns as value. You do not need to synchronize all fields.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyFieldDelimiter</td>
<td>Writes a key delimiter to Redis. For example, if the keyFieldDelimiter is key=key1\0001id, multiple keys need to be concatenated and requires the value. If only one key exists, this configuration item can be ignored.</td>
<td>No</td>
<td>\0001</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in an operation. batchSize can greatly reduce interactions between Data Integration and PostgreSQL over the network, and increases the overall throughput. However, an excessively large value may cause the running process of Data Integration to become Out of memory (OOM).</td>
<td>No</td>
<td>1,000</td>
</tr>
<tr>
<td>expireTime</td>
<td>The Redis value cache expiration time is permanent validity if this configuration item is left empty.</td>
<td>No</td>
<td>0 (0 means the value is permanent validity)</td>
</tr>
<tr>
<td></td>
<td>• seconds: The current time (in seconds) that specifies the time period in which the data expires.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• unixtime: The Unix time calculated in number of seconds from January 1, 1970, and specifies a future time point in which data expires.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If the expiration time is greater than 60<em>60</em>24*30 (30 days), the server identifies the expiration time as the Unix time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>timeout</td>
<td>The time-out (in milliseconds) that was written to Redis.</td>
<td>No</td>
<td>30,000 (This value covers 30 seconds of network breakdown time)</td>
</tr>
<tr>
<td>dateFormat</td>
<td>The time when data is written into Redis in date format: &quot;yyyy-MM-dd HH:mm:ss&quot;.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Parameter type</td>
<td>Valuefield delimiter</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>writeMode</td>
<td>The Redis write mode. Redis supports different value types, including string, list, set, zset, and hash. Redis Writer can write these data types into a Redis instance. The configuration of writeMode varies slightly based on the value type. The writeMode is configured as follows, only one of the following types can be selected when you configure it.</td>
<td>String (string)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>&quot;writeMode&quot; :{&quot;type&quot;: &quot;string&quot;, &quot;name&quot;: &quot;set&quot;, &quot;valueField Delimiter&quot;: &quot;\u0001&quot; }</td>
<td></td>
<td>\u0001</td>
</tr>
<tr>
<td></td>
<td>The write mode when the value type is string.</td>
<td>Value</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>The delimiter between values when values are strings if there are more than two columns of source data in each row (this configuration item can be ignored if only two columns of source data exist: &quot;key&quot; and &quot;value&quot;), for example, value1\u0001value2 \u0001value3.</td>
<td>Required</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Available value: set (store the data, and overwrite this data if it already exists)</td>
<td>Default value</td>
<td>\u0001</td>
</tr>
</tbody>
</table>

Note: Required Yes. Available value: set (store the data, and overwrite this data if it already exists)
- Redis supports different types of values, including string, list, set, zset, and hash. Redis Writer can also write these data types into Redis. However, the writeMode configuration is slightly different from the value type. Only one of the following five data types can be configured for Redis Writer:

■ String (string)

```
"WriteMode":{
  "type": "string",
  "mode": "set",
  "valueFieldDelimiter": "\0001"
}
```

Parameters:

■ type

■ Description: value type: string
■ Required: Yes

■ mode

■ Description: The write mode when the value type is string.
■ Required: Yes. Available value: set (stores data, and overwrites existing data)

■ valueFieldDelimiter

■ Description: The delimiter between values when the values are strings.
■ Required: No
■ Default value: \0001

■ List of strings

```
"writeMode":{
  "type": "list",
  "mode": "lpush|rpush",
  "Maid": "\0001"
}
```
Parameters:

- **type**
  - **Description:** value type: List
  - **Required:** Yes

- **mode**
  - **Description:** The write mode when the value type is list.
  - **Required:** Yes. Available values: lpush (stores data on the far left of list) | rpush (stores data on the far right of list)

- **valueFieldDelimiter**
  - **Description:** The delimiter between value types that are string. For example: value1\u0001value2\u0001value3.
  - **Required:** No
  - **Default value:** \u0001

- **String collection (set)**

```json
"writeMode":{
  "type": "set",
  "name": "set",
  "valueFieldDelimiter": "\u0001"
}
```
Parameters:

- **type**
  - **Description:** value type: set
  - **Required:** Yes

- **mode**
  - **Description:** The write mode when the value type is set.
  - **Required:** Yes. Available value: `sadd` (stores data into set, and overwrites existing data)

- **valueFieldDelimiter**
  - **Description:** The delimiter between values when the value type is string. For example: `value1\u0001value2\u0001value3`.
  - **Required:** No
  - **Default value:** `\u0001`

- **StringCollection (SET)**

**Note:**

If values are Zset data, each row of data source records must follow this rule: With the exception of key, each row can only contain one set of Score and Value. The Score must be located before Value, so the Redis Writer can parse the Score column and Value column.

```json
"writeMode":{
  "type": "zset",
  "mode": "zadd"
}
```
Configuration item descriptions:

- **type**
  - Description: value type: zset
  - Required: Yes;

- **mode**
  - Description: The write mode when values are Zset data.
  - Required: Yes. Available value: zadd (stores data in the Zset sorted set, and overwrites existing data.)

- **Hash (hash)**

  **Note:**
  If values are hashed, each row of data source records must follow this rule:
  With the exception of key, each row only contains one set of parameter and value. The parameter must be located before value, so that Redis Writer can parse the parameter column and the value column.

```json
"writeMode":{
  "type": "hash",
  "mode": "hset"
}
```

**Parameters:**

- **type**
  - Description: value type: hash
  - Required: Yes

- **mode**
  - Description: The write mode when values are hashed.
  - Required: Yes. Optional value: hmset (stores values in the hash sorted set, and overwrites existing data)

  You need to specify one of the data types. If the data type is left empty, the default data type is "string".

  - Required: No
  - Default value: string
Development in Wizard Mode

Currently, development in Wizard Mode is not supported.

Development in Script Mode

Configure Data Synchronization jobs written to Redis. For more information, see parameter descriptions.

```json
{
    "type": "job",
    "version": "2.0", // version number
    "steps": [
        { // The following is a reader template. You can find the corresponding reader plug-in documentation.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        { // The plug-in name
            "stepType": "redis", // plug-in name
            "parameter": {
                "expireTime": { // redis value cache failure time
                    "seconds": 1000
                },
                "keyFieldDelimiter": "\u0001", // key separator written to redis.
                "dateFormat": "yyyy-MM-dd HH:mm:ss", // time format of date when redis is written
                "datasource": "", // Data Source
                "writeMode": { // write mode
                    "mode": "",
                    "valueFieldDelimiter": "", // Value Type
                    "Keyindexes": [ // primary key index
                        0,
                        1
                    ],
                    "batchSize": "1000", // number of records submitted in one batch size
                },
                "name": "Writer",
                "category": "writer"
            }
        },
        { // The setting
            "setting": {
                "errorLimit": { // Number of error records
                    "record": 0
                },
                "speed": {
                    "throttle": false, // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
                    "concurrent": 1, // Number of concurrent tasks
                    "dmu": 1 // DMU Value
                }
            }
        }
    ]
}
```
1.6.5.17 Configure SQL Server Writer

This topic describes the data types and parameters supported by SQL Server Writer and how to configure Writer in both Wizard and Script mode.

The SQL Server Writer plug-in can be used to write data in target tables of the primary SQL Server database. At the underlying implementation level, the SQL Server Writer connects to a remote SQL Server database through Java Database Connectivity (JDBC), and runs the `insert into...` to write data in an SQL Server instance. The data is submitted to the database in batch within the instance.

The SQL Server Writer is designed for Extract, transform, load (ETL) developers to import data from data warehouses to the SQL Server. The SQL Server Writer can also be used as a data migration tool by DBA and other users.

The SQL Server Writer obtains protocol data (`insert into...`) generated by Reader through the Data Integration framework. If the primary key conflicts with the unique index, the data cannot be written in conflicting lines. To improve performance, use `PreparedStatement + Batch` and configure `rewrittenStatements=true` to buffer data to the thread context buffer. Write requests are initiated only when the data volume in the buffer reaches the threshold.

**Note:**

- Data can be written into a target table only when the target table resides in the primary database.
- The task must have the `insert into...` permission. Other permission requirements depend on statements specified in PreSQL and PostSQL when you configure the task.
Type conversion list

The SQL Server Writer supports most data types in the SQL Server. Check whether the data type is supported before using the SQL Server Writer.

The SQL Server writer converts the list of types for SQL Server, as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>SQL server data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Bigint, Int, Smallint, and Tinyint</td>
</tr>
<tr>
<td>Float point</td>
<td>Float, Decimal, Real Numeric</td>
</tr>
<tr>
<td>String type</td>
<td>Char, Nchar, Ntext, Nvarchar, Text, Varchar, Nvarchar (MAX), and Varchar (MAX)</td>
</tr>
<tr>
<td>Date and time type</td>
<td>Date, Time, and Datetime</td>
</tr>
<tr>
<td>Boolean</td>
<td>Bit</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary, Varbinary, Varbinary (max), and Timestamp</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Require</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The data source must be identical to the added data source. Adding data source is supported in Script Mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The name of the selected table that must be synchronized.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>column</td>
<td>The required fields of the target table into which data is written. These fields are separated by commas (,). For example, &quot;column&quot;: [&quot;id&quot;, &quot;name&quot;,&quot;age&quot;]. If you want to write all columns in turn, use the asterisk (<em>) representation. For example: &quot;column&quot;: [&quot;</em>&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement runs before running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statements in Script Mode. For example: clear old data.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement that runs after running the data synchronization task. Currently, you can run only one SQL statement in Wizard Mode, and more than one SQL statement in Script Mode. For example: add a timestamp.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>writeMode</td>
<td>The specified import mode that allows data insertion. insert: If the primary key conflicts with the unique index, the Data Integration determines the data as dirty data, but retains the original data.</td>
<td>No</td>
<td>Insert</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in batch at a time can greatly reduce interactions between Data Integration and SQL Server over the network, and increase the overall throughput. However, an excessively large value may cause the running process of Data Integration to become Out of memory (OOM).</td>
<td>No</td>
<td>1,024</td>
</tr>
</tbody>
</table>
Development in Wizard Mode

1. Choose source

The following is the configuration item descriptions:

Parameters:

- **Data source**: The datasource in the preceding parameter description section. Enter the configured data source name.
- **Table**: The table in the preceding parameter description section. Select the table for synchronization.
- **Before import**: The preSQL in the preceding parameter description section, namely, the SQL statement that runs before the data synchronization task run.
- **After import**: The postSQL in the preceding parameter description section, which is the SQL statement that runs after running the data synchronization task.
- **Primary key conflict**: The writeMode in the preceding parameter description section. You can select the expected import mode.
2. The field mapping is the column in the above parameter description.

The source table field on the left and the target table field on the right are in one-to-one relationships, click Add row to Add A Single field. To delete the current field, click Delete.

- In-row mapping: You can click Enable Same-Line Mapping to create a mapping for the same row. Note that the data type must be consistent.
- Automatic formatting: The fields are automatically sorted based on corresponding rules.

3. Control the tunnel

Parameters:

- Concurrent job count: The maximum number of threads used to concurrently read/write data into the data storage media in a data synchronization task. You can configure a concurrency for the specified task in Wizard mode.
- The maximum number of errors indicates the maximum number of dirty data records.
- Task resource group: The machine on which the task runs. If the number of tasks is large, the default Resource Group is used to wait for a resource. We recommend that you add a Custom Resource Group. For more information, see Add scheduling resources.

Development in Script Mode

For more information on how to configure jobs written to SQL Server, see specific parameter completion in the preceding parameter description section.

```json
{
  "type": "job",
  "version": 2.0
}
```
1.6.5.18 Configure Elasticsearch Writer

This topic describes the data types and parameters supported by Elasticsearch Writer and how to configure Writer in both Wizard and Script mode.

The Elasticsearch is a Lucene-based search and data analysis tool that provides distributed service. Elasticsearch is an open source product based on Apache's open source terms, and is currently a mainstream enterprise-class search engine.
The Elasticsearch core concept that corresponds to core database concepts as follows.

<table>
<thead>
<tr>
<th>Relational DB (Instance)</th>
<th>databases (database)</th>
<th>tables (table)</th>
<th>rows (one row of data)</th>
<th>Columns (one row of data)</th>
<th>innisearch</th>
<th>index</th>
<th>types</th>
<th>documents</th>
<th>Fields</th>
</tr>
</thead>
</table>

There can be multiple indexes (INDEX)/(database) in Elasticsearch, where each index can contain multiple types (type)/(table). Each type can contain multiple document rows, and each document can contain multiple fields (columns). The Elasticsearch Writer plug-in uses Elasticsearch REST API interface to write data that is read from the reader in bulk to Elasticsearch.

**Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>The Elasticsearch URL in the format of <a href="http://xxxx.com:9999">http://xxxx.com:9999</a>.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>accessId</td>
<td>The Elasticsearch user name, which is used for authorizing an Elasticsearch connection.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>accessKey</td>
<td>The password of the Elasticsearch instance.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>index</td>
<td>The index name in Elasticsearch.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>indexType</td>
<td>The index type name in Elasticsearch.</td>
<td>No</td>
<td>Elasticsearch</td>
</tr>
<tr>
<td>cleanup</td>
<td>The parameter that determines if a data exists in an index or has been deleted. The method used to clean data is to delete and rebuild the corresponding index. By default, the value is False which means data in the existing index is retained.</td>
<td>No</td>
<td>False</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of data entries imported in bulk each time.</td>
<td>No</td>
<td>1,000</td>
</tr>
<tr>
<td>trySize</td>
<td>The number of retries after task failure.</td>
<td>No</td>
<td>30</td>
</tr>
<tr>
<td>timeout</td>
<td>The client timeout.</td>
<td>No</td>
<td>600,000</td>
</tr>
<tr>
<td>discovery</td>
<td>When this Node Discovery parameter is enabled, the server list in the client is polled and regularly updated.</td>
<td>No</td>
<td>False</td>
</tr>
<tr>
<td>compression</td>
<td>The parameter that specifies whether compression is enabled for HTTP requests.</td>
<td>No</td>
<td>True</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>multiThread</td>
<td>The HTTP request that specifies if the request is multiple threads.</td>
<td>No</td>
<td>True</td>
</tr>
<tr>
<td>ignoreWriteError</td>
<td>This parameter ignores writing errors and writes without retries.</td>
<td>No</td>
<td>False</td>
</tr>
<tr>
<td>ignoreParseError</td>
<td>This parameter ignores parsing data format errors and continues writes.</td>
<td>No</td>
<td>True</td>
</tr>
<tr>
<td>alias</td>
<td>The Elasticsearch's alias is similar to the database view mechanism, and creates an alias name for the index my_index. This operation is similar to the my_index operation. Configuring the alias means that after completing the data import, an alias is created for the specified index.</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>aliasMode</td>
<td>The modes for adding an alias after data is imported. The modes are append and exclusive.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>settings</td>
<td>If you insert an array type target-side data column, use the specified delimiter (-, -) to separate the data source. For example: The source column data type is a string a, b, c, d that uses the delimiter (-, -). When the string is split into an array [&quot;a&quot;, &quot;b&quot;, &quot;c&quot;, &quot;d&quot;], it is written into the Elasticsearch corresponding to the Filed column.</td>
<td>No</td>
<td>(-, -)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>column</td>
<td>The column used to configure multiple document fields. Each specific field item can configure basic configurations, such as name, type, and more. Available column extension configurations, include Analyzer, Format, and Array. The specific instructions are as follows:</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>The field types supported by Elasticsearch are as follows.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|           | - id  
|           | - string  
|           | - text  
|           | - keyword  
|           | - long  
|           | - integer  
|           | - short  
|           | - byte  
|           | - double  
|           | - float  
|           | - date  
|           | - boolean  
|           | - binary  
|           | - integer_range  
|           | - float_range  
|           | - long_range  
|           | - double_range  
|           | - date_range  
|           | - geo_point  
|           | - geo_shape  
|           | - ip  
|           | - completion  
|           | - token_count  
|           | - array  
|           | - Object  
|           | - nested  
|           | If the column type is text, you can configure the analyzer, norms, and index_options parameters as follows.                                                                                             |          |               |
|           | `{  
|           |   "name": "col_text ",  
|           |   "type": "text",  
|           |   "analyzer": "ik_max_word"  
|           | }  
|           | If the column type is date, you can configure the Format and Timezone parameters. These represent a date serialization format and a time zone, respectively, as follows.                                        |          |               |
|           | `{  
|           |   "name": "col_date ",  
|           |   "type": "date",  
|           |   "format": "yyyy-MM-dd HH:mm:ss",  
|           |   "timezone": "UTC"  
|           | }  

286

Issue: 20191220
Development in Script Mode

The following is an example of a script configuration. For details about the parameter configurations, see the preceding Parameter Description.

```json
{
    "job": {
        "setting": {
            ...
        },
        "content": [
            {
                "reader": {
                    ...
                },
                "writer": {
                    "name": "Elasticsearchwriter",
                    "parameter": {
                        "endpoint": "http://xxxx.com: 9999 "",
                        "accessId": "xxxx",
                        "accessKey": "yyyy",
                        "index": "test-1",
                        "type": "default",
                        "cleanup": true,
                        "settings": {
                            "index": {
                                "number_of_shards": 1,
                                "number_of_replicas": 0
                            },
                            "discovery": false,
                            "batchSize": 1000,
                            "splitter": "",
                            "column": [
                                {
                                    "name": "pk", "type": "id"
                                },
                                {
                                    "name": "col_ip", "type": "ip"
                                },
                                {
                                    "name": "col_double", "type": "double"
                                },
                                {
                                    "name": "col_long", "type": "long"
                                },
                                {
                                    "name": "col_integer", "type": "integer"
                                },
                                {
                                    "name": "col_keyword", "type": "keyword"
                                },
                                {
                                    "name": "col_text", "type": "text",
                                    "analyzer": "ik_max_word"
                                }
                            ],
                            "geo_point": {
                                "name": "col_geo_point",
                                "type": "geo_point"
                            },
                            "date": {
                                "name": "col_date",
                                "type": "date",
                                "format": "yyyy-MM-dd HH:mm:ss"
                            },
                            "nested": {
                                "name": "col_nested1",
                                "type": "nested"
                            },
                            "object": {
                                "name": "col_object1",
                                "type": "object"
                            },
                            "integer_array": {
                                "name": "col_integer_array",
                                "type": "integer",
                                "array": true
                            }
                        }
                    }
            }
        ]
    }
}
```

Note:
Currently, Elasticsearch for the VPC environment can only use custom scheduling resources. If you run the default Resource Group, the network connection will breakdown. For more information on how to add a custom resource group, see #unique_12.

1.6.5.19 Configure LogHub Writer
This topic describes the data types and parameters supported by the LogHub Writer and how to configure the Writer in both Wizard and Script mode.

LogHub Writer uses Java SDK in Log Service (SLS) to push data in DataX Reader to the specified SLS LogHub for other program consumption.

Note:
LogHub cannot realize idempotence. Re-executing the task after FailOver may result in data duplication.

Implementation principles
LogHub Writer uses DataX framework to obtain data generated by the Reader and converts the data types supported by DataX into string data type. When the data size reaches the specified batchSize value, the LogHub Writer uses SLS Java SDK to push all data to LogHub in one batch. By default, 1024 data entries are pushed, and the maximum batchSize value is 4096.

LogHub Writer supports LogHub type conversion as shown in the following table:

<table>
<thead>
<tr>
<th>Internal DataX type</th>
<th>LogHub data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>String</td>
</tr>
<tr>
<td>Double</td>
<td>String</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Date</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>String</td>
</tr>
<tr>
<td>Bytes</td>
<td>String</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>The Log Service address.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>accessKeyId</td>
<td>The accessKeyId for accessing the Log Service instance.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>accessKeySecret</td>
<td>The accessKeySecret for accessing the Log Service instance.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>project</td>
<td>The project name of the target Log Service.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>logstore</td>
<td>The LogStore name of the target Log Service instance.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>topic</td>
<td>Select a topic.</td>
<td>No</td>
<td>Null string</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of data entries that can be pushed at a time.</td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>column</td>
<td>The column name in each data entry.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>

Introduction to Script Mode

Currently, Wizard Mode configuration is not supported. You can click on the link to convert to Script Mode or select import Script Template for development.

Introduction to Script Mode

The following is a script configuration example. For more information about parameters, see the preceding Parameter description section.

```json
{
    "type": "job",
    "version": "2.0", //version number
    "steps": [
        { //The following is a reader template. You can find the corresponding reader plug-in documentations.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        ]
}
```
1.6.5.20 Configure OpenSearch Writer

This topic describes data types and parameters supported by OpenSearch Writer and how to configure Writer in both Wizard and Script mode.

The OpenSearch Writer plug-in is designed to insert or update data in OpenSearch. Data developers can use it to import processed data in OpenSearch and output data by searching. How fast data can be transmitted depends on the account Queries per second (QPS) that corresponds to the OpenSearch table.

Implementation

At the underlying implementation level, OpenSearch Writer provides the publicly available OpenSearch API through OpenSearch.
• OpenSearch v3 uses internal dependent databases, with the following POM: com.aliyun.opensearch aliyun-sdk-opensearch 2.1.3.

Note:
• To use the OpenSearch Writer plug-in, you must use JDK 1.6-32 or later versions. You can view the Java version through java-version.
• Currently, the default resource group does not support connections to the VPC environment because of potential network problems.

Plug-in features

Column order

Because the columns in OpenSearch are unordered, you need to use OpenSearch Writer to write data in strict compliance with the specified column order. If the number of specified columns are less than those in OpenSearch, the redundant columns are set to the default value or null.

For example, if the imported field list contains fields b and c, but the OpenSearch table contains the fields a, b, and c. You can configure the column to "column": ["c ","b"]'). The first two columns in Reader are imported to fields c and b in OpenSearch, and the field a, in which new records are inserted is set to the default value or null.

• How to handle column configuration errors

To ensure data written is reliable, OpenSearch Writer prevents data loss caused by redundant columns that can lead to data quality failure. OpenSearch Writer reports an error when redundant columns are written. If the OpenSearch table contains fields a, b, and c, the OpenSearch Writer generates an error when more than three fields are written by OpenSearch Writer.

• Table configuration precautions

The OpenSearch Writer can only write data from one table at a time.

• Rerun task and failover:

After the task is rerun, the data is automatically overwritten by the ID. Therefore, OpenSearch must contain one ID column. The ID uniquely identifies a record line in OpenSearch. The data is the same as the overwritten unique ID.

• Rerun task and failover:

After a task is rerun, the data is automatically overwritten by the IDs.
OpenSearch Writer supports most OpenSearch data types. Check whether the data type is supported. OpenSearch Writer converts data types in OpenSearch as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>Opensearch data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int</td>
</tr>
<tr>
<td>Float point</td>
<td>Double/Float</td>
</tr>
<tr>
<td>String type</td>
<td>TEXT/Literal/SHORT_TEXT</td>
</tr>
<tr>
<td>Date and time type</td>
<td>Int</td>
</tr>
<tr>
<td>Boolean</td>
<td>Literal</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessId</td>
<td>The Logon ID for Alibaba Cloud.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>accessKey</td>
<td>The Logon Key for Alibaba Cloud.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>host</td>
<td></td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>indexName</td>
<td>The name of the OpenSearch project.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The table for which the data is written into. You cannot enter more than one</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>table because DataX does not support importing multiple tables simultaneously.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>column</td>
<td>The list of imported fields. If you need to import all fields, it can be</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>configured to &quot;column&quot;: [&quot;*&quot;]. Enter the specified columns, if you need to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>insert OpenSearch columns. For example: &quot;column&quot;: [&quot;id&quot;, &quot;name&quot;]. OpenSearch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>supports column filtering and column order changes. For example, you can</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>configure the fields to [&quot;c, b&quot;], if a table has three fields: a, b, and c,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and only fields c and b must be synchronized. The field a is automatically</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>inserted with null values and set to null during import.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of data lines written in a single entry. Data is written into OpenSearch in batches. Typically, the advantage of OpenSearch is query, but has low write Transactions per seconds (TPS) performance. Proceed with the configuration based on the resources applied to your account. For OpenSearch, a single data item size is generally less than 1 MB, and the size of each written data entry is less than 2 MB.</td>
<td>This field is required for a partition table, but is not required if the target table is a non-partition table.</td>
<td>300</td>
</tr>
</tbody>
</table>
| writeMode  | In OpenSearch Writer, "writeMode": "add/update" is configured to ensure the idempotence of write operations.  
  - "add": When a reattempt is made after a failed write attempt, OpenSearch Writer clears the data and imports new data (atomic operation).  
  - "update": It indicates the data is inserted in a modified manner (atomic operation). | Yes | None |

Note: Because batch insert is not an atomic operation in OpenSearch, it may be partially successful. Therefore, writeMode is a key option and currently OpenSearch with version =v3 does not support update**.
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignoreWriteError</td>
<td>This parameter ignores write errors.</td>
<td>No</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>The following is an example of the configuration: &quot;ignoreWriteError&quot;: true.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenSearch performs batch write operations. When ignoreWriteError is enabled,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>all write failures are ignored, but continues other write operations. When</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>this parameter is disabled, an error is returned when a write failure</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>occurs and the task ends. We recommend you use the default value: False, for</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>The OpenSearch version information.</td>
<td>No</td>
<td>v2</td>
</tr>
<tr>
<td></td>
<td>The following is a configuration example: &quot;version&quot;: &quot;v3&quot;. OpenSearch v3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>is more preferable because OpenSearch v2 has multiple push operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>limitations.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Development in Script Mode

Configure the data synchronization job to write data to OpenSearch:

```json
{
  "type": "job",
  "version": "1.0",
  "configuration": {
    "reader": {},
    "writer": {
      "plugin": "opensearch",
      "parameter": {
        "accessId": "*********",
        "accessKey": "*********",
        "host": "http://yyyy.aliyuncs.com",
        "indexName": "datax_xxx",
        "table": "datax_yyy",
        "column": ["appkey", "id", "title", "gmt_create", "pic_default"],
        "batchSize": 500,
        "writeMode": add,
        "version": "v2",
        "ignoreWriteError": false
      }
    }
  }
}
```
1.6.5.21 Configure Table Store (OTS) Writer

This topic describes the data types and parameters supported by Table Store (OTS) Writer and how to configure Writer in both Wizard and Script mode.

Table Store (formerly known as OTS) is a NoSQL database service built-on Alibaba Cloud ApsaraDB Distributed Operating System that allows storage and real-time access of massive structured data. Table Store organizes data into instances and tables. Table Store provides seamless scaling by using data partition and Server Load Balancing (SLB) technology.

In short, the Table Store Writer-Internal connects to the Table Store server through the official Table Store Java Software Development Kits (SDKs), and writes data in the Table Store server through the SDKs. The Table Store Writer has optimized the writing process to include retry upon writing timeout, retry upon writing exception, batch submissions, and other features.

Currently, the Table Store Writer-Internal supports all types of Table Store data and converts data types for Table Store as follows:

- **PutRow**: The PutRow for Table Store API, which is used to insert data in a specified row. If this row does not exist, a new row is added. Otherwise, the original row is overwritten.
- **UpdateRow**: The UpdateRow for Table Store API, which is used to update the data of a specified row. If the row does not exist, a new row is added. Otherwise, the specified column values are added, modified, or deleted based on the request.

Currently, Table Store Writer supports all Table Store data types and converts the Table Store data types as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>Table store data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Integer</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Note:
The Integer category must be configured to Int in Script Mode for it to be converted to Integer type in Table Store. You cannot configure the Integer type in Table Store, an error will occur in the log and lead to task failure.

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name must be identical to the added data source name. Script Mode supports adding data source.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>endPoint</td>
<td>The table store server endpoint. For more information, see Access control.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>accessId</td>
<td>The AccessId of a Table Store instance.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>accessKey</td>
<td>The AccessKey required for accessing Table Store service.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
| instanceName | The name of the Table Store instance.  
An instance is an object for using and managing the Table Store. After Table Store is activated, you need to create an instance through the console, and then create and manage tables in the instance. An instance is the basic unit for Table Store resource management. The Table Store controls access to applications and measures resources on an instance-level. | Yes      | None          |
| table     | Selects the table name for extraction. You can enter only one table name. Multi-table synchronization is not required for Table Store. | Yes      | None          |

- **primaryKey**
  - Primary key information of the Table Store. The field information is described with JSON arrays. The Table Store is a NoSQL system, so the corresponding field name must be specified when the Table Store Writer imports data.
  - Required: Yes.
  - The PrimaryKey of Table Store only supports String and Int data types, therefore only these two data types can be entered in Table Store Writer.
DataWorks

Data Aggregation / 1 Workbench

Data synchronization system supports data type conversions, so Table Store Writer
can convert the non-String and non-Int data source. Conﬁguration example:
"primaryKey" : [
{"name":"pk1", "type":"string"},
],
• column

- Description: The column name set for synchronization in the conﬁgured table
. The ﬁeld information is described with JSON arrays.

- Required: Yes.

- By default, this ﬁeld is not speciﬁed.

The format is as follows:

{"name":"col2", "type":"INT"},
The parameter "name" speciﬁes the Table Store column name to be written, and "

type" speciﬁes the data type to be written. The data types supported by Table Store,
include String, Int, Double, Bool, and Binary.

Constants, functions, or custom statements are not supported during writing.
• writeMode

- Description: The write mode. The following three modes are supported:

• Single row operation

GetRow: Read data from a single row.
PutRow: PutRow for Table Store API, which is used to insert data to
a specified row. If this row does not exist, a new row is added.
Otherwise, the original row is overwritten.
UpdateRow: UpdateRow for Table Store API, which is used to update the
data of a specified row. If the row does not exist, a new row is added
. Otherwise, the values of the specified columns are added, modified,
or deleted as request.

Issue: 20191220

297


DeleteRow: Delete a row.

- **Batch operation**

BatchGetRow: Read data from multiple rows.

- **Read range**

GetRange: Read table data within a certain range.

  - **Required:** Yes
  - By default, this field is not specified.

Development in Wizard Mode

**Currently, development in Wizard Mode is not supported.**

Development in Script Mode

**Configure a job to write data to Table Store as follows:**

```json
{
    "type": "job",
    "version": 2.0", // version number
    "steps": [
        { //The following is a reader template. You can find the
corresponding reader plug-in documentations.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        { //plug-in name
            "stepType": "ots",
            "parameter": {
                "datasource": "", // Data Source
                "column": [ // Field
                    { "name": "columnname1", // field name
                        "type": "INT" // data type
                    },
                    { "name": "columnname2 ",
                        "type": "STRING"
                    },
                    { "name": "columnname3 ",
                        "type": "double"
                    },
                    { "name": "columnname4 ",
                        "type": "BOOLEAN"
                    },
                    { "name": "columnname5 ",
                        "type": "BINARY"
                    }
                ],
            }
        }
    ]
}
```
"writeMode": "insert", //Write mode
"table": ", // table name
"primaryKey": primary key information for [// table
store
    {
        "name": "pk1",
        "type": "STRING"
    },
    {
        "name": "pk2",
        "type": "INT"
    }
},
"name": "Writer",
"category": "writer"
],
"setting": {
    "errorLimit": {
        "record": "0" //Number of error records
    },
    "speed": {
        "throttle": false, //False indicates that the traffic is
        not throttled and the following throttling speed is invalid. True
        indicates that the traffic is throttled.
        "concurrent": "1", //Number of concurrent tasks
        "dmu": 1 // DMU Value
    }
},
"order": {
    "hops": [
        {
            "from": "Reader",
            "to": "Writer"
        }
    ]
}
}

1.6.5.22 Configure RDBMS Writer
This topic describes the data types and parameters supported by the RDBMS Writer.

The RDBMS Writer plug-in provides the capability to write data into the target table of the master RDBMS database. At the underlying implementation level, the RDBMS Writer connects to a remote RDBMS database through JDBC, and runs the SQL statement insert into...to write data into RDBMS. The RDBMS Writer is a relational database write plug-in for generic purposes, allowing you to add any relational database write support by registering database drivers or other methods.

RDBMS Writer is designed for Extract, transform, load (ETL) developers to import data from data warehouses to RDBMS. The RDBMS Writer can also be used as a data migration tool by DBA and other users.
Implementation principles

RDBMS Writer uses the DataX framework to obtain the protocol data generated by the Reader. Then it connects to a remote RDBMS database through JDBC, and runs the SQL statement `insert into...` to write data into RDBMS.

Function description

Configuration sample

- Configure a job for writing data into RDBMS as follows.

```json
{
  "job": {
    "setting": {
      "speed": {
        "channel",
      }
    },
    "content": [
      {
        "reader": {
          "name": "streamreader",
          "parameter": {
            "Column": [
              {
                "value": "DataX",
                "type": "string",
              },
              {
                "value": 19880808,
                "type": "long",
              },
              {
                "value": "1988-08-08 08:08:08",
                "type": "date",
              },
              {
                "doc_value": true,
                "type": "bool",
              },
              {
                "value": "test",
                "type": "bytes",
              }
            ],
            "sliceRecordCount": 1000
          }
        },
        "writer": {
          "name": "RDBMS Writer",
          "parameter": {
            "connection": [
              {
                "jdbcUrl": "jdbc:dm://ip:port/database",
                "table": [
                  "table"
                ]
              }
            ]
          }
        }
      }
    }
  }
}
```
Parameter description

- `jdbcUrl`
  - Description: The JDBC connection information of the opposite-end database. The `jdbcUrl` format is based on the RDBMS official specification, which allows you to enter the URL attachment control information. Note that databases have different JDBC formats, and DataX selects the appropriate database driver for data reading based on a specific JDBC format.
    - **DM**: `jdbc:dm://ip:port/database`
    - **DB2 format**: `jdbc:db2://ip:port/database`
    - **PPAS format**: `jdbc:edb://ip:port/database`

How to add database support using RDBMS Writer:

- Enter the corresponding directory of the RDBMS Writer. This `${DATAX_HOME}` is the main directory of DataX, that is, `${DATAX_HOME}/plugin/writer/RDBMS Writer`.
- Go to the plugin.json file under the RDBMS Writer directory and register your database driver into the file, which will keep the database driver in the drivers array. The RDBMS Writer plug-in will dynamically select an appropriate database driver to connect the database during task execution.
Go to libs subdirectory under the directory of RDBMS Writer and keep your database driver in the libs subdirectory.

```
$tree
|-- libs
|   |-- Dm7JdbcDriver16.jar
|   |-- commons-collections-3.0.jar
|   |-- commons-io-2.4.jar
|   |-- commons-lang3-3.3.2.jar
|   |-- commons-math3-3.1.1.jar
|   |-- datax-common-0.0.1-SNAPSHOT.jar
|   |-- datax-service-face-1.0.23-20160120.024328-1.jar
|   |-- db2jcc4.jar
|   |-- druid-1.0.15.jar
|   |-- edb-jdbc16.jar
|   |-- fastjson-1.1.46.sec01.jar
|   |-- guava-r05.jar
|   |-- hamcrest-core-1.3.jar
|   |-- jconn3-1.0.0-SNAPSHOT.jar
|   |-- logback-classic-1.0.13.jar
|   |-- logback-core-1.0.13.jar
|   |-- plugin-rdbms-util-0.0.1-SNAPSHOT.jar
|   |-- slf4j-api-1.7.10.jar
|   |-- plugin.json
|   |-- plugin_job_template.json
|   `-- RDBMS Writer-0.0.1-SNAPSHOT.jar
```

- **Required:** Yes
- **By default, this field is not specified.**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>The data source user name.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>password</td>
<td>The password corresponding to the specified user name for the data source.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Parameters</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>table</td>
<td>The target table name. If the table schema information is inconsistent with the user name in the preceding configuration, enter the table information in the schema.table format.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>column</td>
<td>The column name set to be synchronized in the configured table is separated by commas (,). We strongly do not recommend you use the default column configuration.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>PreSQL</td>
<td>The SQL statement that runs before the data synchronization task run. Currently, you can run only one SQL statement. For example: clear old data.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>PostSQL</td>
<td>The SQL statement that runs before the data synchronization task run. Currently, you can run only one SQL statement. For example: add a timestamp.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Parameters</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in one operation. Setting this parameter can greatly reduce interaction between DataX and RDBMS over the network, and increase the overall throughput. However, an excessively large value may cause the running process of DataX to become Out of Memory (OOM).</td>
<td>No</td>
<td>1024</td>
</tr>
</tbody>
</table>

**Type conversion**

RDBMS Reader supports most generic relational database types, such as numbers and characters. Check whether the data type is supported and select a reader based on the specified database.

### 1.6.5.23 Configure Stream Writer

This topic describes the data types and parameters supported by Stream Writer and how to configure Writer in Script Mode.

The Stream Writer plug-in allows you to read data from the Reader and print data on the screen or discard data. It is primarily applied to testing, such as for data synchronization performance and basic functions.

**Parameter description**

- **Print**
  - Description: Whether to print the output data on the screen.
  - Required: No
  - Default value: True
Development in Wizard Mode

Currently, development in Wizard Mode is not supported.

Development in Script Mode

Configure a job to read data from the Reader and print data on the screen as follows:

```json
{
    "type": "job",
    "version": 2.0, // version number
    "steps": [
        // The following is a reader template. You can find the corresponding reader plug-in documentations.
        { // The following is a reader template. You can find the corresponding reader plug-in documentations.
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        { // plug-in name
            "stepType": "stream", // plug-in name
            "parameter": {
                "print": false, // do you want to print output to the screen?
                "fieldDelimiter": "," // Delimiter of each column
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": { // Number of error records
            "record": "0"
        },
        "speed": { // False indicates that the traffic is not throttled and the following throttling speed is invalid. True indicates that the traffic is throttled.
            "throttle": false,
            "concurrent": "1", // Number of concurrent tasks
            "dmu": "1" // DMU Value
        }
    },
    "order": {
        "hops": [
            { // from:"Reader",
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```
1.6.5.24 Configure HybridDB for MySQL Writer

This topic describes the data types and parameters supported by HybridDB for MySQL Writer and how to configure it in both Wizard and Script modes.

HybridDB for MySQL Writer writes data into a HybridDB for MySQL database. At the underlying implementation level, HybridDB for MySQL Writer connects to a remote HybridDB for MySQL database through JDBC, and runs the `INSERT INTO...` or `REPLACE INTO...` SQL statement to write data into the database. Internally, data is submitted to the database in batches, and therefore the database must use the InnoDB engine.

---

**Note:**

You must configure a data source before configuring HybridDB for MySQL Writer. For more information, see [Configure a HybridDB for MySQL data source](#).

HybridDB for MySQL Writer is designed for ETL developers to import data from data warehouses to HybridDB for MySQL. HybridDB for MySQL Writer can also be used as a data migration tool by DBA and other users. HybridDB for MySQL Writer obtains protocol data generated by a reader through the Data Integration framework. The generated protocol data varies with the `writeMode` attribute that you have configured.

---

**Note:**

The task must have the `INSERT INTO...` or `REPLACE INTO...` permission. Whether other permissions are required depends on the SQL statements specified in the `preSql` and `postSql` attributes in the configured task.

---

**Type conversion list**

Similar to HybridDB for MySQL Reader, HybridDB for MySQL Writer supports most data types in HybridDB for MySQL. Before configuring HybridDB for MySQL Writer, check whether the data type is supported.

HybridDB for MySQL Writer converts the data types in HybridDB for MySQL as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>HybridDB for MySQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int, Tinyint, Smallint, Mediumint, Bigint, and Year</td>
</tr>
<tr>
<td>Type classification</td>
<td>HybridDB for MySQL data type</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Float</td>
<td>Float, Double, and Decimal</td>
</tr>
<tr>
<td>String</td>
<td>Varchar, Char, Tinytext, Text, Mediumtext, and Longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date, Datetime, Timestamp, and Time</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Tinyblob, Mediumblob, Blob, Longblob, and Varbinary</td>
</tr>
</tbody>
</table>

### Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name must be identical to the added data source name. Script Mode supports adding data sources.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The destination table name.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
| writeMode | The Write Mode, which can be set to insert or replace.  
- REPLACE INTO…: When there are no primary key or unique index conflicts, the action is the same as that of INSERT INTO. If a conflict occurs, the fields in new rows replace all fields in original rows.  
- INSERT INTO…: If a primary key or unique index conflict occurs, data cannot be written into the conflicting rows, and is classified as dirty data. | No | Insert |
<p>| column | The required destination table fields into which data is written. These fields are separated with commas (,). For example, &quot;column&quot;: [&quot;id&quot;,&quot;name&quot;,&quot;age&quot;]. If you want to write all columns in turn, use the asterisk (<em>). For example: &quot;column&quot;: [&quot;</em>&quot;]. | Yes | None |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>preSql</td>
<td>The SQL statement that runs before running the data synchronization task. For example, you can clear old data before data synchronization. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statements in Script Mode.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement that runs after running the data synchronization task. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statements in Script Mode.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted at a time. This parameter can greatly reduce the interaction frequency between Data Integration and HybridDB for MySQL on the network, and increase the overall throughput. However, an excessively large value may lead to OOM during the data synchronization process.</td>
<td>No</td>
<td>1024</td>
</tr>
</tbody>
</table>
Development in Wizard Mode

1. Specify data sources

Configure the source and destination of data for a synchronization task as follows.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>The datasource parameter in the preceding parameter description. Select the configured data source.</td>
</tr>
<tr>
<td>Table</td>
<td>The table parameter in the preceding parameter description. Select the destination table.</td>
</tr>
<tr>
<td>Statements Run Before Import</td>
<td>The preSQL parameter in the preceding parameter description. Enter the SQL statement that runs before running the data synchronization task.</td>
</tr>
<tr>
<td>Statements Run After Import</td>
<td>The postSql parameter in the preceding parameter description. Enter the SQL statement that runs after running the data synchronization task.</td>
</tr>
<tr>
<td>Primary Key Violation</td>
<td>The writeMode parameter in the preceding parameter description. Select the expected write mode.</td>
</tr>
</tbody>
</table>
2. Configure mappings of fields (the column parameter in the preceding parameter description).

Each source table field on the left maps a destination table field on the right. To add a mapping, and click Add. To delete the current mapping, move the cursor over a line and click Delete.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Fields with the Same Name</td>
<td>Click Map Fields with the Same Name to establish a mapping between fields with the same name. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Map Fields in the Same Line</td>
<td>Click Map Fields in the Same Line to establish a mapping for the same row. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Remove Mappings</td>
<td>Click Remove Mappings to remove established mappings.</td>
</tr>
<tr>
<td>Auto Layout</td>
<td>The fields are automatically sorted based on specified rules.</td>
</tr>
</tbody>
</table>
### 3. Configure channel control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMU</td>
<td>The unit that measures the resources, including CPU, memory, and network resources consumed by Data Integration. A DMU represents the minimum operating capability of a Data Integration task, that is, the data synchronization processing capability given to the limited CPU, memory, and network resources.</td>
</tr>
<tr>
<td>Concurrent Jobs</td>
<td>The maximum number of threads used to concurrently read data from the source or write data into the data storage media in a data synchronization task. In Wizard Mode, you can configure the concurrency for a task on the wizard page.</td>
</tr>
<tr>
<td>Dirty Data Records Allowed</td>
<td>The maximum number of errors or dirty data records allowed.</td>
</tr>
<tr>
<td>Task Resource Group</td>
<td>The machines on which tasks are run. If a large number of tasks run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group. Currently, a custom resource group can be added only in China (Hangzhou) and China (Shanghai). For more information, see #unique_12.</td>
</tr>
</tbody>
</table>

**Development in Script Mode**

The following code is an example of configuration in Script Mode. For more information about parameters, see the preceding parameter description.

```json
{
    "type": "job",
    "steps": []
    {
        "parameter": {},
        {
            "parameter": {
                "postSql": [],// The SQL statement to be run after the data synchronization task is run.
            }
        }
    }
}```
1.6.5.25 Configure HybridDB for PostgreSQL Writer

This topic describes the data types and parameters supported by HybridDB for PostgreSQL Writer and how to configure it in both Wizard and Script modes.

HybridDB for PostgreSQL Writer writes data into a HybridDB for PostgreSQL database. At the underlying implementation level, HybridDB for PostgreSQL Writer connects to a remote HybridDB for PostgreSQL database through JDBC, and runs SELECT statements to extract data from the database. On the public cloud, RDS provides the HybridDB for PostgreSQL storage engine.

**Note:**

312

Issue: 20191220
You must configure a data source before configuring HybridDB for PostgreSQL Writer. For more information, see "Configure a HybridDB for PostgreSQL data source."

In short, HybridDB for PostgreSQL Writer connects to a remote HybridDB for PostgreSQL database through a JDBC connector, generates SELECT statements based on the configuration, and sends the statements to the remote database. Then, HybridDB for PostgreSQL Writer assembles SQL execution results into abstract datasets in custom data types of Data Integration, and passes the datasets to the downstream writer.

- HybridDB for PostgreSQL Writer concatenates the configured table, column, and WHERE undefined information into SQL statements, and sends the statements to the HybridDB for PostgreSQL database.
- HybridDB for PostgreSQL Writer sends the configured query SQL information to HybridDB for PostgreSQL database.

**Note:**

**Type conversion list**

HybridDB for PostgreSQL Writer supports most data types in HybridDB for PostgreSQL. Check whether a data type is supported before configuring HybridDB for PostgreSQL Writer.

HybridDB for PostgreSQL Writer converts the data types in HybridDB for PostgreSQL as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>HybridDB for PostgreSQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>Bigint, Bigserial, Integer, Smallint, and Serial</td>
</tr>
<tr>
<td>Double</td>
<td>Double precision, Money, Numeric, and Real</td>
</tr>
<tr>
<td>String</td>
<td>Varchar, Char, Text, Bit, and Inet</td>
</tr>
<tr>
<td>Date</td>
<td>Date, Time, and Timestamp</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Bytes</td>
<td>Bytea</td>
</tr>
</tbody>
</table>

**Note:**

- Only the preceding field types are supported.
To convert Money, Inet, and Bit data types, you need to use syntax, such as `a_int::varchar`.

### Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. The name must be identical to the added data source name. Script Mode supports adding data sources.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The name of the destination table.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>writeMode</td>
<td>The Write Mode, which can be set to insert data. Insert: If a primary key conflict or unique index conflict occurs, Data Integration determines the data as dirty data, and retains the original data.</td>
<td>No</td>
<td>Insert</td>
</tr>
<tr>
<td>column</td>
<td>The destination table fields into which data needs to be written. These fields are separated with commas (,). For example, &quot;column&quot;: [&quot;id&quot;, &quot;name&quot;, &quot;age&quot;]. If you want to write all columns in turn, use the asterisk (<em>). For example: &quot;column&quot;: [&quot;</em>&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement that runs before running the data synchronization task. For example, you can clear old data before data synchronization. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statements in Script Mode.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement runs after running the data synchronization task. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement in Wizard Mode, and multiple SQL statements in Script Mode.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted in a batch. This parameter can greatly reduce the interaction frequency between Data Integration and HybridDB for PostgreSQL on the network, and increase the overall throughput. However, an excessively large value may lead to OOM during the data synchronization process.</td>
<td>No</td>
<td>1024</td>
</tr>
</tbody>
</table>

**Development in Wizard Mode**

1. **Specify data sources.**

   Configure the data source and destination for a synchronization task.

   ![Data Source Configuration](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>The datasource parameter in the preceding parameter description. Select the configured data source.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>The table parameter in the preceding parameter description. Select the destination table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Before Import</td>
<td>The preSQL parameter in the preceding parameter description. Enter the SQL statement that runs before running the data synchronization task.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>After Import</td>
<td>The postSQL parameter in the preceding parameter description. Enter the SQL statement that runs after running the data synchronization task.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Configure mappings of the fields (the column parameters in the preceding parameter description).

Each source table field on the left maps a destination table field on the right. To add a mapping, click Add. To delete the current mapping, move the cursor over a line and click Delete.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Fields with the Same Name</td>
<td>Click Map Fields with the Same Name to map fields with the same name. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Map Fields in the Same Row</td>
<td>Click Map Fields in the Same Row to map the same row. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Remove Mappings</td>
<td>Click Remove Mappings to remove established mappings.</td>
</tr>
<tr>
<td>Auto Layout</td>
<td>The fields are automatically sorted based on specified rules.</td>
</tr>
</tbody>
</table>

3. Configure channel control

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent Jobs</td>
<td>The maximum number of threads used to concurrently read data from the source or write data into the data storage media in a data synchronization task. In Wizard Mode, you can configure the concurrency for a task on the wizard page.</td>
</tr>
<tr>
<td>Dirty Data Records Allowed</td>
<td>The maximum number of errors or dirty data records allowed.</td>
</tr>
<tr>
<td>Task Resource Group</td>
<td>The machines on which tasks are run. If a large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group. For more information, see #unique_12.</td>
</tr>
</tbody>
</table>
Development in Script Mode

```json
{
   "type": "job",
   "steps": [
      {
         "parameter": {},
         "name": "Reader",
         "category": "reader"
      },
      {
         "parameter": {
            "postSql": [], // The SQL statement to be run after the data synchronization task is run.
            "datasource": "test_004", // The data source name.
            "column": [ // The destination table columns.
               "id",
               "name",
               "sex",
               "salary",
               "age"
            ],
            "table": "public.person", // The destination table name.
            "preSql": [] // The SQL statement to be run before the data synchronization task is run.
         },
         "name": "Writer",
         "category": "writer"
      }
   ],
   "version": "2.0", // The version number.
   "order": {
      "hops": [
         {
            "from": "Reader",
            "to": "Writer"
         }
      ]
   },
   "setting": {
      "errorLimit": { // The maximum number of errors allowed.
         "record": ""
      },
      "speed": {
         "concurrent": 6, // The number of concurrent threads.
         "throttle": false // Indicates whether to throttle the transmission rate.
      }
   }
}
```

1.6.5.26 Configure POLARDB Writer

This topic describes the data types and parameters supported by POLARDB Writer and how to configure it in both wizard and script modes.

POLARDB Writer writes data into a POLARDB database. At the underlying implementation level, POLARDB Writer connects to a remote POLARDB database.
through JDBC, and runs the `INSERT INTO...` or `REPLACE INTO...` SQL statement to write data into the database. Internally, data is submitted to the database in batches, and therefore the database must use the InnoDB engine.

**Note:**
You must configure a data source before configuring POLARDB Writer. For more information, see [Configure a POLARDB data source](#).

POLARDB Writer is designed for ETL developers to import data from data warehouses to POLARDB. POLARDB Writer can also be used as a data migration tool by DBA and other users. POLARDB Writer obtains protocol data generated by a reader through the Data Integration framework. The generated protocol data varies with the `writeMode` attribute that you have configured.

**Note:**
The task shall at least have the `INSERT INTO...` or `REPLACE INTO...` permission. Whether other permissions are required depends on the SQL statements specified in the `preSql` and `postSql` attributes when you configure the task.

**Type conversion list**

Similar to POLARDB Reader, POLARDB Writer supports most data types in POLARDB. Check whether a data type is supported before configuring POLARDB Writer.

POLARDB Writer converts the data types in POLARDB as follows:

<table>
<thead>
<tr>
<th>Type classification</th>
<th>POLARDB data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Int, Tinyint, Smallint, Mediumint, Bigint, and Year</td>
</tr>
<tr>
<td>Float</td>
<td>Float, Double, and Decimal</td>
</tr>
<tr>
<td>String</td>
<td>Varchar, Char, Tinymce, Text, Mediumtext, and Longtext</td>
</tr>
<tr>
<td>Date and time</td>
<td>Date, Datetime, Timestamp, and Time</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Binary</td>
<td>Tinymce, Mediumtinymce, Blob, Longtinymce, and Varbinary</td>
</tr>
</tbody>
</table>
## Parameter description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>The data source name. It must be identical to the data source name added. Adding data sources is supported in script mode.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>table</td>
<td>The name of the destination table.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>writeMode</td>
<td>The write mode, which can be set to insert or replace.</td>
<td>No</td>
<td>insert</td>
</tr>
<tr>
<td></td>
<td>• REPLACE INTO...: If no primary key conflict or unique index conflict occurs, the action is the same as that of INSERT INTO. If a conflict occurs, the fields in new rows replace all fields in original rows.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• INSERT INTO...: If a primary key conflict or unique index conflict occurs, data cannot be written into the conflicting rows and is regarded as dirty data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• INSERT INTO table (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE...: If no primary key conflict or unique index conflict occurs, the action is the same as that of INSERT INTO. If a conflict occurs, the fields in new rows replace the specified fields in original rows.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>column</td>
<td>The fields of the destination table into which data needs to be written. These fields are separated with commas. For example, &quot;column&quot;: [&quot;id&quot;,&quot;name&quot;,&quot;age&quot;]. If you want to write all columns in turn, use the asterisk (<em>), for example, &quot;column&quot;: [&quot;</em>&quot;].</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>preSql</td>
<td>The SQL statement to be run before the data synchronization task is run. For example, you can clear old data before data synchronization. Currently, you can run only one SQL statement in wizard mode, and multiple SQL statements in script mode.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>postSql</td>
<td>The SQL statement to be run after the data synchronization task is run. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement in wizard mode, and multiple SQL statements in script mode.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>batchSize</td>
<td>The number of records submitted at a time. This attribute can greatly reduce the frequency of interaction between Data Integration and POLARDB on the network, and increase the overall throughput. However, an excessively large value may lead to OOM during the data synchronization process.</td>
<td>No</td>
<td>1024</td>
</tr>
</tbody>
</table>
Development in wizard mode

1. Specify data sources

Configure the source and destination of data for a synchronization task.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>The datasource attribute in the preceding parameter description. Select the data source that you have configured.</td>
</tr>
<tr>
<td>Table</td>
<td>The table attribute in the preceding parameter description. Select the destination table.</td>
</tr>
<tr>
<td>Statements Run Before Import</td>
<td>The preSql attribute in the preceding parameter description. Enter the SQL statement that is run before the data synchronization task is run.</td>
</tr>
<tr>
<td>Statements Run After Import</td>
<td>The postSql attribute in the preceding parameter description. Enter the SQL statement that is run after the data synchronization task is run.</td>
</tr>
<tr>
<td>Primary Key Violation</td>
<td>The writeMode attribute in the preceding parameter description. Select the expected write mode.</td>
</tr>
</tbody>
</table>
2. Configure mappings of fields (the column attribute in the preceding parameter description).

Each source table field on the left maps a destination table field on the right. You can click Add to add a mapping or move the cursor over a line and click Delete to delete the current mapping.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Fields with the Same Name</td>
<td>Click Map Fields with the Same Name to establish a mapping between fields with the same name. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Map Fields in the Same Line</td>
<td>Click Map Fields in the Same Line to establish a mapping for the same row. Note that the data type must be consistent.</td>
</tr>
<tr>
<td>Remove Mappings</td>
<td>Click Remove Mappings to remove mappings that have been established.</td>
</tr>
<tr>
<td>Auto Layout</td>
<td>The fields are automatically sorted based on specified rules.</td>
</tr>
</tbody>
</table>
3. Configure channel control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent Jobs</td>
<td>The maximum number of threads used to concurrently read data from the source or write data into the data storage media in a data synchronization task. In wizard mode, you can configure the concurrency for a task on the wizard page.</td>
</tr>
<tr>
<td>Dirty Data Records Allowed</td>
<td>The maximum number of errors or dirty data records allowed.</td>
</tr>
<tr>
<td>Task Resource Group</td>
<td>The machines on which tasks are run. If a large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group. For more information, see Add task resources.</td>
</tr>
</tbody>
</table>

Development in script mode

The following code is an example of configuration in script mode. For more information about attributes, see the preceding parameter description.

```json
{
    "type": "job",
    "steps": [
        {
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "parameter": {
                "postSql": [], // The SQL statement to be run after the data synchronization task is run.
                "datasource": "test_005", // The data source name.
                "column": [ // The destination table columns.
                    "id",
                    "name",
                    "age",
                    "sex",
                    "salary",
                    "interest"
                ]
            }
        }
    ]
}
```
"writeMode": "insert"// The write mode.
"batchSize": 256,// The number of records submitted at a time.
"encoding": "UTF-8",// The encoding format.
"table": "POLARDB_person_copy",// The destination table name.
"preSql": []// The SQL statement to be run before the data synchronization task is run.
],
"name": "Writer",
"category": "writer"
}
,"version": "2.0",// The version number.
"order": {
"hops": [
{
"from": "Reader",
"to": "Writer"
}
],
"setting": {
"errorLimit": {// The maximum number of errors allowed.
"record": ""
},
"speed": {
"concurrent": 6,// The number of concurrent threads.
"throttle":false,// Indicates whether to throttle the transmission rate.
"dmu": 6// The DMU value.
}
}
}

1.6.6 Optimizing configuration

This topic describes how to adjust DMU and concurrent configuration of synchronization jobs for optimized maximum synchronization speed. The data synchronization speed is influenced by factors, including differences between speed-limit jobs and not-speed-limit jobs, and precautions for custom resource groups.

DataWorks Data Integration supports real-time, offline data interconnection between any data sources in any location and network environment. It is a comprehensive full-stack data synchronization platform allows you to copy dozens of TBs data between various cloud and local data storage media.

The super fast data transmission performance and interconnection between more than 400 pairs of heterogeneous data sources that helps users focus on core big data issues. The service can be used to design advanced analysis solutions with deep insight in all data.
Factors affecting data synchronization speed

The factors that affect data synchronization speed as follows.

- **Source-side data sources**
  - Database performance: The performance of CPU, memory module, SSD, network, and hard disk.
  - Concurrency: A high data source concurrency results in a high database workload.
  - Network: The bandwidth (throughput) and network speed. Typically, a database with better performance can tolerate a higher concurrency. Therefore, the data synchronization job can be configured for high-concurrency data extraction.

- **Synchronous task configuration for Data Integration**
  - Synchronization speed: Determines whether a synchronization speed limit is set.
  - DMU: The resources used for running the synchronization task.
  - Concurrency: The maximum number of threads that can be used to read/write data from the data source to the target data source at the same time in one synchronization task.
  - The wait resource.
  - Bytes setting: If the Bytes limit is set to 1,048,576, and the network is slow, the data transmission times out before completion. We recommend you set a lower Bytes value.
  - Whether to create an index for query statements.

- **Objective to end Data Source**
  - Performance: The performance of the CPU, memory module, SSD, network, and hard disk.
  - Load: The high database load that affects data write efficiency.
  - Network: The bandwidth (throughput) and network speed.

You need to monitor and optimize the performance, load, and network of the original data source and destination databases. The following mainly describes how to set core configurations of a synchronization task in Data Integration.
DMU

- **Configuration**

  A data synchronization task can run with single or multiple DMUs. In Wizard mode, you can configure a maximum of 20 DMUs for a task. The following is an example of how to set the number of DMUs in Script mode:

  ```json
  "Setting ":{
    "Speed ":{
      "dmu": 10
    }
  }
  
  Note:
  Although, you can configure more than 20 DMUs with a script, the system still has certain resource limitations. We recommend that you do not assign too many resources.

- **DMU writing synchronization speed factors**

  The DMU represents the resource capability, and the synchronization task is configured with a higher DMU. You can allocate more resources without increasing the synchronization task speed. Speed optimization requires combining the concurrency with the DMU ratio. For example, a synchronization task that configures 3 concurrencies requires 3 DMU, and the synchronization speed is 10 Mb/s. Currently, the number of 3 concurrent resources required is 3 DMU, and the task does not require more resources. Increasing the DMU does not, therefore, increase the synchronization task speed.

Concurrent

- **Configuration**

  In Wizard Mode, configure a concurrency for the specified task on the wizard page. The following is an example of configuring the number of concurrency with Script Mode.

  ```json
  "Setting ":{
    "Speed ":{
      "concurrent": 10
    }
  }
  ```
• The relationship between concurrency and DMU

A higher concurrency requires more DMUs. When network conditions and data source performances are good, more DMUs and higher concurrency will result in better synchronization speed.

- To ensure that a task in Wizard mode can be successfully executed in high concurrency, the maximum concurrency allowed cannot exceed the number of DMUs set. For example, we do not recommend that you configure more than 10 concurrency threads when the number of DMUs is set to 10.
- When a high concurrency is set, you need to consider data source capabilities in the reading and writing ends. Excessive concurrency may affect the source database performance. Therefore, you need to tune the database.
- In Script Mode you can set a high concurrency. However, the number of DMUs that are provided for a task are limited. Do not set an excessively high concurrency.

Speed limit

By default, throttling is disabled after the Beta phase of Data Integration ends. In a synchronization task, data is synchronized at the maximum speed supported by the concurrency and DMUs configured for that task. Because extremely fast synchronization may overstress the database and affect production, Data Integration allows you to limit synchronization speed and optimize configuration as required. We recommend that the maximum configured speed limit cannot exceed 30 MB/s when this option is enabled. The following is an example for configuring the speed limit in Script Mode, when the transmission bandwidth is 1 MB/s:

```
"Setting ":{
  "Speed ":{
    "throttle": true // Throttling enabled.
    "mbps": 1, // Synchronization speed
  }
}
```

Note:

• Throttling is disabled when it is set to False. You do not need to configure the mbps parameter.
• The traffic measured value is a Data Integration metric that does not represent actual NIC traffic. Generally, the NIC traffic is two to three times that of the channel traffic, which depends on the serialization of data storage system.

• A semi-structured Single file does not have shard key concept. Multiple files can set the maximum job rate to increase the synchronization speed, however, the maximum job rate is related to the number of files. For example, there are n files with maximum job rate limit set to n Mb/s, then if you set n + 1 Mb/s or sync at n Mb/s speed. If you set the speed to n-1 Mb/s, then the synchronization is performed at n-1 Mb/s speed.

• The table splitting can be performed at the set maximum job rate, only when a maximum job rate and a shard key are configured for a relational database. Relational databases only supports numeric shard keys, but Oracle databases support both Numeric and String shard keys.

Cases of slow data synchronization

Synchronization tasks remain in the waiting status when using public scheduling (WAIT) resources

• Related examples are as follows

When you test synchronization tasks in DataWorks, multiple tasks remain in the waiting status and an internal system error occurs.

It takes 800 seconds to synchronize a task from RDS to MaxCompute using default resource groups, but the log shows that the task runs for only 18 seconds and stops. Other synchronization tasks with hundreds of data entries also remain in the waiting status.

The waiting log is displayed as follows:

```
2017-01-03 07: 16: 54: State: 2 (wait) | Total: 0r 0b | speed: 0r/s 0b/S | error: 0r 0b | stage: 0.0%
```

• Solution

In this case, public scheduling resources are used. The capability is limited because they share many projects instead of two or three tasks of a single user. A 10-second task is extended to 800 seconds because the required resources were insufficient and must be waited when you run the task.

If you have strict requirements for synchronization speed and waiting time, we recommend starting synchronization tasks during non-busy hours. Typically
, synchronization tasks are concentrated between 00:00 and 03:00. You can perform synchronization tasks in other time except from the aforesaid period to avoid resource waiting.

Accelerate tasks of synchronizing data in multiple tables to the same table

- Related examples are as follows:

  Synchronization tasks are serialized to synchronize the tables of multiple data sources to the same table, but the synchronization duration can take a long time.

- Solution

  To start multiple write data tasks in the same database simultaneously, pay attention to the following:

  - Ensure the load capacity of the destination database is sufficient to prevent improper runs.
  - When you configure workflow tasks. Select a single task node and configure database or table shard tasks, or set multiple nodes to run concurrently in the same workflow.
  - If the synchronization tasks encounter resource waiting (WAIT) during runs, run them during non-rush hours for high execution priority.

No index added while using the SQL WHERE clause

- Related examples are as follows:

  The executed SQL statement as follows:

  ```sql
  select bid, inviter, uid, createTime from `relatives` where createTime >= '2016-10-23 00:00:00' and createTime < '2016-10-24 00:00:00';
  ```

  If the query statement execution started at 2016-10-25 11:01:24.875 Beijing Time (UTC+8), then the return query result started at 2016-10-25 11:11:05.489 Beijing Time (UTC+8). The synchronization program waited the database to return the SQL query result, and MaxCompute waited for a long time to start.

- Cause analysis:

  When the WHERE statement was executed, the createTime column was not indexed and full-table scanning was enforced.
· Solution

We recommend that you add an index to the scan column, if you want to use the SQL WHERE clause.
2 Data source configuration

2.1 Data source isolation

Data source isolation can be used to isolate data of the development environment from data of the production environment for workspaces in standard mode.

If a data source is configured in both the development and production environments, you can use data source isolation to isolate the data source in the development environment from that in the production environment.

Note:
Currently, only workspaces in standard mode support data source isolation.

When you configure a data synchronization task, the data source in the development environment is used. When you submit the data synchronization task to the production environment for running, the data source in the production environment is used. To submit a task to the production environment for scheduling, you must configure a data source in both the development and production environments. A data source must have the same name in the development and production environments.

Data source isolation has the following impacts on workspaces:

- Workspaces in basic mode: The functions and configuration pages of data sources are the same as those before the data source isolation feature is added. For more information, see Data source configuration.
- Workspaces in standard mode: The Applicable Environment parameter is added on the configuration pages of data sources.
- Workspaces upgraded from the basic mode to the standard mode: During the upgrade, you will be prompted to upgrade data sources. After the upgrade, the data sources in the development environment are isolated from those in the production environment.
<table>
<thead>
<tr>
<th>Page element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Migrate Tables from Data Stores | Click Migrate Tables from Data Stores to go to the Batch Sync page.  
  **Note:**  
  You can select a data source on the Batch Sync page only after the data source is configured in both the development and production environments and has passed the connectivity test. |
<p>| Add Connections | Currently, you can add only multiple MySQL, SQL Server, or Oracle data sources at a time. The template contains the data source type, data source name, data source description, environment type (0 for development and 1 for production), and URL. You can download the template, configure multiple data sources in the template, and upload the template to add the data sources at a time. On the page for adding multiple data sources at a time, details about the data sources will be displayed. |</p>
<table>
<thead>
<tr>
<th>Page element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Add Data Source       | • If the environment is set to development for a data source, you can select the data source when creating a data synchronization node. The node task can be executed in the development environment. However, you cannot submit the node task to the production environment for running.  
  • If the environment is set to production for a data source, you can use the data source only in the production environment. You cannot select the data source when creating a data synchronization node. |
<p>| Note:                 | A data source must have the same name in the development and production environments.                                                                                                                        |
| Applicable Environment| For a workspace in basic mode, this column is not displayed. For a workspace in standard mode, this column is displayed to show the environment of each data source.                                          |</p>
<table>
<thead>
<tr>
<th>Page element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Actions      | • Migrate Tables Configuration: This button is displayed for a data source in the development environment and can be clicked when the data source is configured in both the development and production environments.  
• Add Data Source: This button is displayed if a data source is not configured in an environment.  
• Edit and Delete: The two buttons are displayed for a data source that has been configured in an environment.  

- Before deleting a data source from both the development and production environments, check whether the data source is used by any synchronization task in the production environment. The delete operation cannot be rolled back. After the data source is deleted, you cannot select it when configuring a synchronization task in the development environment.  

If a synchronization task in the production environment uses the data source, the synchronization task cannot be executed after the data source is deleted. Delete the synchronization task before deleting the data source.  

- Before deleting a data source from the development environment, check whether the data source is used by any synchronization task in the production environment. The delete operation cannot be rolled back. After the data source is deleted, you cannot select it when configuring a synchronization task in the development environment.  

If a synchronization task in the production environment uses the data source, you cannot obtain metadata when editing the synchronization task after the data source is deleted. However, the synchronization task can be executed.  

- Before deleting a data source from the production environment, check whether the data source is used by any synchronization task in the production environment. If you select the data source when configuring a synchronization task in the development environment, you cannot submit the task for publishing in the production environment after the data source is deleted.  

334
### 2.2 Configure AnalyticDB for MySQL 2.0 data source

This topic describes how to configure an AnalyticDB for MySQL 2.0 data source. AnalyticDB for MySQL 2.0 allows you to write data to AnalyticDB for MySQL 2.0, but does not allow you to read data from it. AnalyticDB for MySQL 2.0 supports data integration in wizard and script mode.

**Procedure**

1. Log on to the *DataWorks console* as an administrator and click Enter Workspace in the actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.
4. In the Create Data Source dialog box, set the data source type to AnalyticDB for MySQL 2.0.
5. Complete the AnalyticDB for MySQL 2.0 data source configuration items.

**Configurations:**

- **Name:** The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description:** A brief description of the data source cannot exceed 80 characters in length.
- **Link URL:** The AnalyticDB for MySQL 2.0 URL. Format: serverIP:Port.
- **Schema:** The AnalyticDB for MySQL 2.0 schema information.
- **AccessID/AccessKey:** The access key (AccessKeyID and AccessKeySecret) is equivalent to the login password.

6. Click Test Connectivity.
7. After completing the test connectivity, click Complete.

The test connectivity determines if the information entered is valid.
Next step

For more information on how to configure the AnalyticDB for MySQL 2.0 Writer plug-in, see [Configure AnalyticDB for MySQL 2.0 Writer](#).

2.3 Configure SQL Server data source

This topic describes how to configure SQL server data source. The SQL server data source allows you to read and write data to SQL server instances, and supports configuring synchronization tasks in wizard and script mode.

**Note:**

Currently, only SQL Server 2005 or later versions are supported. If the SQL server is in a VPC environment, please note the following issues:

- Create an on-premise SQL server data source

  - Test connectivity is not supported, but the synchronization of task configuration is supported. You can synchronize task configurations by clicking OK when creating the data source.
  
  - You must use a custom scheduled Resource Group to run corresponding synchronization tasks, make sure the Custom Resource Group can connect to the on-premise database. For more information, see [#unique_13](#) and [#unique_14](#).

- SQL server data sources created with RDS

  You do not need to select a network environment, the system will automatically determine the data source based on the information entered for the RDS instance.

**Procedure**

1. Log on to the DataWorks console as an administrator, and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type SQL Server in the new dialog box.
5.Configure the SQL Server data source information separately.

The SQL server data source types are categorized into Alibaba Cloud Database (RDS), Public Network IP Address, and Non-public Network IP Address. You can select the data type based on your requirements.

Consider the new data source of SQL Server > Alibaba Cloud Database (RDS) type.

**Configurations:**
- **Type:** ApsaraDB for Relational Database Server (RDS).
- **Name:** The name must start with a letter or underscores (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
• Description: A brief description of the data source that cannot exceed 80 characters in length.

• RDS instance ID: You can view the RDS instance ID in the RDS console.

- RDS instance buyer ID: You can view the buyer's information under the RDS console security settings.

• Username and password: The user name and password for database connection.

Note:
You need to add a RDS whitelist before connecting to the database.

Consider a data source with a new SQL Server > Public Network IP Address type.

Add SQL Server Connection

* Connect To: Connection string mode (data integrated network can be directly connected)

* Connection Name: Enter a name.

Description:

* Applicable: Development Production

Environment

* JDBC URL: jdbc:sqlserver://ServerIP:Port;DatabaseName=Database

* Username:

* Password:

Test Connection: Test Connection

Ensure that the database is available.
Ensure that the data sent to and from the database can pass through the firewall.
Ensure that the database domain name can be resolved.
Ensure that the database has been started.

Configurations:

- Type: The SQL Server Data Source with a public IP address.
- Name: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- Description: A brief data source description that cannot exceed 80 characters in length.
- JDBC URL: JDBC connection information in the form of jdbc:sqlserver://ServerIP:Port;DatabaseName=Database.
- Username and password: The user name and password used to connect to the database.
Consider a data source with a newSQL Server > Public Network IP Address type.

**Configurations:**

- **Type:** A data source without a public IP address.
- **Name:** The name must start with a letter or underscore (_) and can be 1 to 60 characters in length. It can contain letters, numbers, or underscores (_).
- **Description:** A brief description of the data source. It must be 1 to 80 characters in length.
- **Resource group:** It is used to run synchronization tasks, and generally multiple machines can be bound when you add a resource group. For more information, see `#unique_12`.
• JDBC URL: The JDBC connection information in the form of jdbc:sqlserver://ServerIP:Port;DatabaseName=Database.
• Username and password: The user name and password used to connect to the database.

6. Click Test Connectivity.
7. When the test connectivity is passed, click Complete.

Connectivity test description
• The connectivity test is available in the classic network configuration, identify whether the input JDBC URL, user name, and password are correct.
• Private network and no public network IP address, currently does not support data source connectivity test, click OK.

Next step
For more information on how to configure the SQL Server Writer plug-in, see Configure SQL Server Writer.

2.4 Configure MongoDB data source
This topic describes how to configure MongoDB data sources in DataWorks. MongoDB, is one of the world's most popular document-based NoSQL databases following Oracle and MySQL. The MongoDB data source allows you to read/write data to MongoDB, and supports configuring synchronization tasks in Script Mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type MongoDB in the new data source dialog box.
5. Complete the MongoDB data source item configuration.

MongoDB data source types are categorized into ApsaraDB and On-Premise Database Public Network IP Address.

- ApsaraDB: These databases generally use classic networks. The classic network does not support cross-region connections.
- User-created databases with public IP addresses: These databases generally use public networks that may incur certain costs.

Consider a data source with a new MongoDB > ApsaraDB type.

**Configurations:**

- **Data Source Type:** Select the data source type "MongoDB: Alibaba Cloud database".
Note:
If you have not granted the default role data integration system permission, the primary account is required to go to RAM for role authorization and then refresh the page.

- Name: A name must start with a letter or underscores (_) and can be 60 characters in length. It can contain letters, numbers, and underscores (_).
- Description: A brief description of the data source that does not exceed 80 characters in length.
- Region: Refers to the selected region when purchasing MongoDB.
- Instance ID: You can view the MongoDB instance ID in the MongoDB console.
- Database name: You can create a new database in the MongoDB console, configure the corresponding data name, user name, and password.
- Username and password: The user name and password used for the database connection.

The following is an example of a data source with a new MongoDB > On-Premise Database with Public Network IP Address.
Configurations:

- **Type**: Select the data source type "MongoDB: User-created database with a public IP address".
- **Name**: A name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that does not exceed 80 characters in length.
- **Visit address**: The format is host:port.
- **Add visit address**: Add an access address in the format of host:port.
- **Database name**: The database name mapped to the data source.
- **Username and password**: The user name and password used to connect to the database.
6. Click Test Connectivity
7. If the connectivity passed the test, click Complete.

Note:

- A MongoDB cloud database in a VPC environment is added with a public network IP address data source type and saved.
- Currently, the VPC network does not support connectivity tests.

For more information on how to configure the MongoDB Writer plug-in, see Configure MongoDB Writer.

2.5 DataHub data source

This topic describes how to configure a DataHub data source. DataHub provides a comprehensive data import solution that accelerates massive data computing. DataHub data source allows other data sources to write data to DataHub and supports the Writer plug-in.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type DataHub in the new dialog box.
5. Complete the DataHub data source individual items configurations.

Configurations:

- **Name**: The name must start with a letter or underscore (_), and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **DataHub endpoint**: By default, this parameter is read-only and is automatically read from the system configuration.
- **DataHub project**: The DataHub project ID.
- **AccessID/AcessKey**: The access key (AccessKeyID and AccessKeySecret) is equivalent to the logon password.

6. Click Test Connectivity.

7. When the connectivity passes the test, click Complete.

   Provides connectivity test capabilities to determine if the information entered is correct.
Next step

For more information on how to configure the Oracle Writer plug-in, see Configure DataHub Writer.

2.6 Configure the DM data source

This topic describes how to configure the DM data source. The DM relational database data source provides the capability to read/write data to DM databases, and supports configuring synchronization tasks in wizard and script modes.

Procedure

1. Log on to the DataWorks console as an administrator (primary account) and click Enter Workspace from the Actions column of the relevant project in the Project List.

2. Select Data Integration in the top navigation bar. Click Data Source from the left-side navigation pane.
3. Click **New Source** in the supported data source window.

4. In the new dialog box, select the DM data source type.
5. Complete the DM data source information items configurations.

Select either of the following data source types as required when creating a DM data source:

- The New DM Data Sources with public IP address

Parameters:

- **Type**: DM Data Sources with a public IP address.
- **Name**: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **JDBC URL**: In the format of jdbc:mysql://ServerIP:Port/Database.
- **Username and password**: The user name and password used for connecting to the database.

- **New DM Data Sources without public IP address**

```
Add DM Connection

* Connect To: User-Created Data Store without Public IP Addresses
  Synchronization is supported for data sources of this type only through custom resource groups. [Learn more.]

* Connection Name: [Enter a name]
  Description: [Add Description]

* Applicable: [Select Development] [Select Production]
  Environment

* Resource Groups: [Select Resource Group]
  Add Resource Group

* JDBC URL: [jdbc:dm://ServerIP:Port/Database]

* Username: [Add Username]
* Password: [Add Password]

Test Connection: [Test Connection]
  You cannot perform connectivity tests for data stores without public IP addresses.
```

**Parameters:**

- **Type**: DM data sources without public network IP address. Selecting this data source type requires the use of custom scheduling resources for synchronization. You can click Help manual for details.

- **Name**: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).

- **Description**: A brief description of the data source that does not exceed 80 characters in length.
- Resource Group: It is used to run synchronization tasks, and generally you can bound multiple machines when adding a resource group. For more information, see `unique_12`.
- JDBC URL: In the format of `jdbc:mysql://ServerIP:Port/Database`.
- Username and password: The user name and password to connect to the database.

6. (Optional) Click Test Connectivity to test the connectivity after entering all the required field information.

7. When the connectivity has passed the test, click Complete.

   Provides test connectivity capability to determine if the information entered is correct.

Connectivity test description

- The connectivity test is available in the classic network to identify whether the entered JDBC URL, user name, and password are correct.
- Currently, VPC and data source types without public IP addresses do not support connectivity tests. As a result, click Confirm.

2.7 Configure DRDS data sources

This topic describes how to configure DRDS data sources. The DRDS data source allows you to read/write data to DRDS, and supports configuring synchronization tasks in wizard and script mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.

2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. In the new data source dialog box, select the data source type DRDS.
5. Enter the DRDS data source configuration items.

Configurations:

- **Name**: The name must start with a letter or underscore (_), and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that does not exceed 80 characters in length.
- **JDBC URL**: The JDBC URL format is: `jdbc:mysql://serverIP:Port/database`.
- **Username and password**: The user name and password used for database connection.

6. Click Test Connectivity
7. When the connectivity has passed the test, click Complete.

The DRDS data source provides test connectivity capability for verifying the entered information validity.

Connectivity test description

- The connectivity test is available in the classic network environment to identify whether the entered JDBC URL, user name, and password are valid.
- Currently, the private network or IP addresses without public network, and data source connectivity tests are not supported. Click OK.

Next step

For more information on how to configure the DRDS Writer plug-in, see Configure DRDS Writer.

2.8 Configure FTP data source

This topic describes how to configure the FTP data source. The FTP data source allows you to read/write data to FTP, and supports configuring synchronization tasks in wizard and script mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type FTP in the new data source pop-up window.
5. Complete the FTP data source information items configuration.

You can create either one of the following two FTP data sources:

- **FTP data sources with public IP address**

  ![Add FTP Connection](image)

  **Configurations:**
  
  - **Type:** A FTP data source with a public IP address.
  - **Name:** The name must start with a letter or underscores (_), and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
  - **Description:** A brief description of the data source that cannot exceed 80 characters in length.
  - **Protocol:** Currently, only supports FTP and SFTP.
  - **Host:** The FTP host IP address.
- **Port:** If you select the FTP protocol, the default port is 21. If SFTP is selected, port 22 is used by default.

- **Username and password:** The account and password for accessing the FTP service.

**FTP data sources without public IP address**

Add FTP Connection

- **Connect To:** User-Created Data Store without Public IP Addresses

  Synchronization is supported for data sources of this type only through custom resource groups. [Learn more.](#)

- **Connection Name:** Enter a name.

- **Description:**

- **Environment**

- **Resource Groups:** Select a resource group.

- **Protocol:** FTP, SFTP

- **Host:** Enter the FTP host.

- **Port:** 21

- **Username:**

- **Password:**

- **Test Connection:** You cannot perform connectivity tests for data sources.

**Configurations:**

- **Data source type:** The FTP data sources without a public IP address. This data source type must use custom scheduling resources so that it can synchronize data. For details, click Help Manual.
- Data source name: The name must start with a letter or underscore (\_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (\_).

- Data source description: A brief description of the data source that does not exceed 80 characters in length.

- Resource Group: The resource group is used to run synchronization tasks. You can bind multiple machines when you add a resource group. For details, see Add scheduling resources.

- Protocol: Currently, only FTP and SFTP are supported.

- Host: The FTP host IP address.

- Port: If you select the FTP protocol, the port defaults to 21. If SFTP is selected, the port 22 is used by default.

- Username and password: The account and password for accessing the FTP service.

6. Click Test Connectivity

7. When the test connectivity is finished, click Complete.

The test connectivity capability provided determines if the information entered is correct.

Connectivity test description

- The connectivity test is available in the classic network to identify whether the entered host, port, user name, and password information is correct.

- The data source connectivity test is currently not supported by the VPC network, and you can click Confirm.

Next step

For more information on how to configure the FTP Writer plug-in, see Configure FTP Writer.
2.9 Configuring HDFS data source

This topic describes configuring a HDFS data source. HDFS is a distributed file system that allows you to read/write data to HDFS, and supports configuring synchronization tasks in Script Mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace from the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source on the supported data source pop-up window.

4. In the new data source pop-up window, and select the data source type HDFS.
5. Configure HDFS data sources information separately.

**Configurations:**

- **Name:** A name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description:** A brief description of the data source and cannot exceed 80 characters in length.
- **defaultFS:** The node address of nameNode in the format of hdfs://ServerIP:Port.

6. Click Test Connectivity

7. When the connectivity has passed the test, click Complete.

Configure the HDFS data source that provides test connectivity capability to determine if the entered information is correct.

**Connectivity test description**

- The connectivity test is available in the classic network to verify whether the entered JDBC URL, user name, and password are valid.
- Currently, the VPC network does not support data source connectivity tests. Click OK.
Next step

For more information on how to configure the HDFS Writer plug-in, see Configure HDFS Writer.

2.10 Add LogHub data source

This topic describes how to add a LogHub data source. The LogHub is a data hub, and LogHub data source allows you to read/write data to LogHub. LogHub supports Reader and Writer plug-ins.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type LogHub in the new dialog box.
5. Configure individual information items for the LogHub data source.

Configurations:

- **Name**: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It contains letters, numbers, and underscores (_).
- **Data source description**: A brief description of the data source that does not exceed 80 characters in length.
- **LogHub Endpoint**: Generally, the LogHub Endpoint format is in http://cn-shanghai.log.aliyun.com. For more information, see service entrance.
- **Project**: The project name.
- **AccessKey ID/AccessKey Secret**: The *AccessKey* (AccessKeyID and AccessKeySecret) is equivalent to the logon password.

6. Click Test Connectivity.

7. When the connectivity has passed the test, click Complete.

The connectivity test is provided to identify whether the entered AccessKey project information is correct.
Next step

For more information on how to configure LogHub reader/writer, see Configure LogHub Reader and Configure LogHub Writer.

2.11 Configure MaxCompute data source

This topic describes how to configure a MaxCompute data source. The MaxCompute (formerly known as ODPS) provides a comprehensive data import solution that accelerates massive data computing. As a data hub, the MaxCompute data source allows you to read/write data on MaxCompute, and supports reader and writer plug-ins.

![Note:]
By default, a data source (odps_first) is generated for each project. The MaxCompute project name is the same as that for the current project computing engine.

The AccessKey of the default data source can click on the user information in the upper right corner and change the AccessKey information modification, but it should be noted that:

1. You can only switch AccessKeys between primary accounts.
2. When switching there cannot be any tasks in operation whether it is data integration or data development and all other tasks related to DataWorks. MaxCompute data sources you added manually can use the RAM user AccessKey.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type MaxCompute (ODPS) in the new window.
5. **Complete the MaxCompute data source configurations.**

Configurations:

- **Data source name:** The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Data source description:** A brief description of the data source that does not exceed 80 characters in length.
- **MaxCompute endpoint:** By default, the MaxCompute endpoint is read-only. The value is automatically read from the system configuration.
- **MaxCompute project name:** The corresponding MaxCompute project indicator.
- **AccessID/AccessKey:** The AccessKey (AccessKeyId and AccessKeySecret) is equivalent to the logon password.
6. Click Test Connectivity.

7. When the connectivity has passed the test, click Complete.

The provided connectivity test can identify whether the entered project and AccessKey information is valid.

Next step

For more information on how to configure the MaxCompute Writer plug-in, see Configure MaxCompute Writer.

2.12 Configure Memcache data source

This topic describes how to configure Memcache data source. The Memcache (formerly known as OCS) data source provides the ability to write data from other data sources to Memcache, and supports configuring synchronization tasks in script mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.

2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select Memcached as the data source type in the new dialog box.
5. Complete the Memcache data source configuration.

Configurations:

- **Name**: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **Type**: Select Memcache as the data source type.
- **Proxy Host**: The corresponding Memcache proxy.
- **Port**: The corresponding Memcache port. The default port is 11211.
- **Username and password**: The database user name and password.

6. Click Test Connectivity

7. When the connectivity has passed the test, click Complete.

The Memcache provides test connectivity capabilities to determine whether the entered information is valid.

Next step

For more information on configure the Memcache Writer plug-in, see *Configure Memcache (OCS) Writer*. 
2.13 Configure MySQL data source

This topic describes how to configure the MySQL data source. The MySQL data source allows you to read/write data on MySQL, and supports configuring synchronization tasks in wizard and script mode.

Note:

If you are using MySQL in a VPC environment, you need to be aware of the following issues.

- On-premise MySQL data source
  - Does not support test connectivity, but supports synchronization task configuration. You can configure synchronization task by clicking Confirm when creating the data source.
  - You must use a custom scheduled Resource Group to run the corresponding synchronization tasks, make sure the Custom Resource Group can connect to the on-premise database. For more information, see `#unique_13` and `#unique_14`.

- MySQL data sources created with RDS
  
  You do not need to select a network environment, the system will automatically determine the network environment based on information entered for the RDS instance.

Procedure

1. Log on to the `DataWorks console` as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Log on to the `DataWorks console` as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
3. Click Data Integration in the top navigation bar to go to the Data Source page.
4. **Click Add Data Source in the supported data source pop-up window.**

5. **Select the data source type MySQL in the new dialog box.**
6. Complete the MySQL data source information items configuration.

MySQL Data source types are divided in the Alibaba Cloud Database (RDS), the Public Network IP Address and the Non-Public Network IP Address.

Consider a data source for the new MySQL > Alibaba Cloud Database (RDS) type.

**Configurations:**

- **Type:** Currently, the selected data source type MySQL > Alibaba Cloud Database (RDS).
- **Name:** A name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
• **Description:** A brief description of the data source that does not exceed 80 characters in length.

• **RDS Instance ID:** You can go to the RDS console to view the RDS instance ID.

![RDS Instance ID](image)

- **Instance ID:** rm-n8j4j4z14
- **Instance Region and Zone:** China North 1 (Qingdao) Zone B

• **RDS instance buyer ID:** You can view information in the RDS console security settings.

![Security Settings](image)

- **Login Account:** [Redacted]
- **Account ID:** [Redacted]
- **Registration Time:** May 2, 2017 4:47:00 PM

• **Username and password:** The user name and password used to connect to the database.

**Note:**
You need to add an RDS whitelist before connection. For more information, see #unique_74.

Consider a data source for the new MySQL > Public Network IP Address type as an example.

**Configurations:**

- **Type:** A new MySQL data source with a public IP address.
- **Name:** A name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It must contain letters, numbers, and underscores (_).
- **Description:** A brief description of the data source that does not exceed 80 characters in length.
- **JDBC URL:** The format is `jdbc:mysql://serverIP:Port/database`.

Ensure that the database is available.
Ensure that the data sent to and from the database can pass through the firewall.
Ensure that the database domain name can be resolved.
Ensure that the database has been started.
- **Username and password**: The user name and password used for connecting to the database.

For example, a data source with a new MySQL > Non-Public Network IP Address type.

**Configurations:**

- **Data source type**: The data source without a public IP address.
- **Data source name**: A data source name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It must contain letters, numbers, and underscores (_).
- **Data source description**: A brief description of the data source that does not exceed 80 characters in length.
DataWorks

Data Aggregation / 2 Data source configuration

- Resource group: A group used to run synchronization tasks, and generally multiple machines can be bound when you add a resource group. For more information, see #unique_12.

- JDBC URL: The format is jdbc:mysql://serverIP:Port/Database.

- Username and password: The user name and password used for connecting the database.

7. Click Test Connectivity.

8. Click OK after the connectivity has passed the test.

Connectivity test description

- The connectivity test is available in the classic network environment for verifying whether the entered JDBC URL, user name, and password are valid.

- Currently, the private network and no public network IP address, data source connectivity test is not supported. Click OK.

Next step

For more information on how to configure the MySQL Writer plug-in, see Configure MySQL Writer.

2.14 Configure Oracle data source

This topic describes how to configure an Oracle data source. The Oracle data source allows you to read/write data on Oracle, and supports configuring synchronization tasks in wizard and script mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.

2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source on the supported data source pop-up window.

4. Select the data source type Oracle in the new data source dialog box.

5. Configure each Oracle data source information item.

   Oracle Data source types are categorized into Connection string mode (data integrated network can be directly connected) and User-Created Data Store
without Public IP Addresses, and you can select source types based on your requirements.

For example, a data source that adds a new Oracle > Connection string mode (data integrated network can be directly connected) type.

Configurations:

- **Type**: An Oracle data source with a public IP address.
- **Name**: The name must start with letters or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that does not exceed 80 characters in length.
- **JDBC URL**: The JDBC URL format is: jdbc:oracle:thin:@serverIP:Port:Database.
• Username and password: The user name and password used for connecting to the database.

Consider a data source that adds a new Oracle > User-Created Data Store without Public IP Addresses type.

Configurations:

• Type: When there are no public network IP addresses, this data source type requires custom scheduling resources for synchronization. You can click the Help Manual to view it.

• Name: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
• Description: A brief description of the data source that does not exceed 80 characters in length.
• JDBC URL: The format of the JDBC URL is: jdbc:oracle:thin:@host:port:SID or jdbc:oracle:thin:@//host:port/service_name.
• Username and password: The user name and password used for connecting to the database.

6. Click Test Connectivity
7. When the connectivity has passed the test, proceed by clicking Complete.

Connectivity test description

• The connectivity test is available in the classic network environment to identify whether the entered JDBC URL, user name, and password are correct.
• Currently, does not support private network, IP addresses without public network and data source connectivity, proceed by clicking OK.

Next step

For more information on how to configure Oracle Writer plug-in, see Configuring Oracle Writer.

2.15 Configure OSS data source

This topic describes how to configure an Object Storage Service (OSS) data source. OSS is a massive, secure, and highly reliable cloud storage service offered by Alibaba Cloud.

Note:

• If you want to learn more about OSS products, see the OSS Product Overview.
• The OSS Java SDK can be found in the Alibaba Cloud OSS Java SDK.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click **New Source** in the supported data source pop-up window.

4. Go to the new dialog box, and select the data source type **OSS**.
5. Complete the OSS Data Source configuration items.

Configurations:

- Name: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- Description: A brief description of the data source that does not exceed 80 characters in length.
- Endpoint: The OSS endpoint information format is `http://oss.aliyuncs.com`. It is the endpoint of the OSS service and the Region. When you visit an endpoint in a different region, you need to enter different domain names.

Note:
The correct endpoint format is `http://oss.aliyuncs.com`. You need to add the bucket value in the point number format before the OSS connects to `http`
DataWorks - Data Aggregation / 2 Data source configuration

For example, `http://xxx.oss.aliyuncs.com` can pass connectivity tests, but will report errors during synchronization.

- **Bucket**: The OSS instance bucket. The bucket is a storage space and serves as the container for storing objects. You can create multiple buckets and add multiple files to each bucket. You can search for corresponding files in the data synchronization task through the entered bucket, and file searching is unavailable for buckets that have not been added.

- **AccessID/AccessKey**: The `AccessKey` (AccessKeyId and AccessKeySecret) is equivalent to the logon password.

6. Click Test Connectivity
7. When the connectivity has passed the test, click Complete.

Connectivity test description

- The connectivity test is available in classic network to identify whether the entered Endpoint and AccessKey information is correct.
- The data source connectivity test is currently not supported by the VPC network, and you can click OK.

Next step

The next topic describes how to configure the OSS writer plug-in. For more information, see Configure OSS Writer.

### 2.16 Configure Table Store (OTS) data source

This topic describes how to configure Table Store (OTS) data source. Table Store is a NoSQL database service built on Alibaba Cloud’s Apsara distributed file system, enabling you to store and access massive volumes of structured data in real time.

**Note:**

For more information about Table Store, see [Table Store Product Overview](#).

**Procedure**

1. Click Data Integration in the top navigation bar to go to the Data Source page.
2. Click New Source on the supported data source pop-up window.

3. Select the data source type Table Store (OTS) in the new dialog box.
4. Complete the Table Store data source configuration.

**Configurations:**

- **Name:** The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description:** A brief description of the data source that does not exceed 80 characters in length.
- **Endpoint:** The endpoint format of the Table Store server [http://yyy.com](http://yyy.com). For more information, see **Endpoint**.
- **Table Store Instance ID:** The Instance ID corresponding to the Table Store service.
- **AccessID/AccessKey:** The **AccessKey** (AccessKeyId and AccessKeySecret) is equivalent to the logon password.

5. Click Test Connectivity

6. When the connectivity passed the test, click Complete.
Connectivity test description

- The connectivity test is available in the classic network to identify whether the entered endpoint or AccessKey information is correct.
- The VPC network currently does not support data source connectivity test. Click OK.

2.17 Configure PostgreSQL data source

This topic describes how to configure a PostgreSQL data source. The PostgreSQL data source allows you to read/write data on PostgreSQL, and supports configuring synchronization tasks in wizard and script mode.

Note:
If the PostgreSQL is in a VPC environment, you need to note the following issues:

- On-premise PostgreSQL data source
  - The on-premise PostgreSQL does not support test connectivity, but supports synchronization task configuration. You can synchronize task configurations by clicking OK, when creating the data source.
  - You must use a custom scheduled Resource Group to run the corresponding synchronization tasks, ensure the Custom Resource Group can connect to the on-premise database. For more information, see #unique_13 and #unique_14.

- PostgreSQL data sources created with RDS
  You do not need to select a network environment, the system automatically selects the network environment based on the RDS instance information.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type PostgreSQL in the new dialog box.
5. Complete the PostgreSQL data source individual information items configuration.

PostgreSQL data source types are categorized into Apsara DB for RDS, Public Network IP Address, and Non-Public Network IP Address. You can select the data source type based on the situation.

The following is an example of how to add a new PostgreSQL > Apsara DB for RDS type.

Configurations:

• Type: Apsara DB for RDS.
• Name: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. The name can contain letters, numbers, and underscores (_).

• Description: A brief description of the data source that cannot exceed 80 characters in length.

• RDS instance ID: You can view the RDS instance ID in the RDS console.

The following figure is an example of a data source that adds a PostgreSQL > With a Public Network IP Address type.
Configurations:

- **Type**: A PostgreSQL data source with a public IP address.
- **Name**: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It must contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **JDBC URL**: The JDBC URL format is: `jdbc:postgresql://ServerIP:Port/database`.
- **Username and password**: The user name and password used for connecting to the database.

The following is an example of new PostgreSQL > Data Source Without Public Network IP Address type.
Configurations:

- **Type**: A PostgreSQL data source without a public IP address.
- **Name**: The name must start with a letter or underscore (\_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (\_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **Resource Group**: The resource used to run synchronization tasks. Typically, you can bound multiple machines when you add a resource group. For more information, see *Add scheduling resources*.
- **JDBC URL**: The JDBC URL format is: `jdbc:mysql://ServerIP:Port/database`. 

Ensure that the database is available.
• Username and password: The user name and password used for database 
  connection.

6. Click Test Connectivity

7. When the connectivity has passed the test, click Complete.

Connectivity test description

• The connectivity test is available in the classic network to verify whether the 
  entered JDBC URL, user name, and password are valid.
• Currently, private network and IP address without public network does not 
  support data source connectivity test. Click OK.

Next step

For more information on how to configure the PostgreSQL Writer plug-in, see 
Configure PostgreSQL Writer.

2.18 Configure Redis data source

This topic describes how to configure a Redis data source. Redis is a document- 
based NoSQL database that provides persistent memory database services. Based 
on its highly reliable active/standby hot backup architecture and seamlessly 
scalable cluster architecture, this service can meet high read/write performance 
and flexible capacity configuration requirements of businesses. The Redis 
data source allows you to read/write data to Redis, and supports configuring 
synchronization tasks in Script Mode.

Procedure

1. Log on to the DataWorks console as an administrator and click Enter Workspace in 
   the Actions column of the relevant project in the Project List.
2. Click Data Integration in the top navigation bar to go to the Data Source page.
3. Click New Source in the supported data source pop-up window.

4. Select the data source type Redis in the new dialog box.
5. Complete the Redis data source configuration items.

The Redis data source type is categorized into ApsaraDB for RDS and Public Network IP Address On-Premise Database.

- ApsaraDB for RDS: These databases generally use classic networks. You cannot connect cross-region classic networks, only networks in the same region can connect.
- User-created databases with public IP addresses: Generally, these databases use public networks, which may cause you to incur certain costs.

The following figure is an example of adding aRedis > ApsaraDB RDS type.

![Add Redis Connection](image)

**Configurations:**

- **Type:** Currently, the selected data source type is Redis > Apsara DB RDS.

**Note:**
If you have not authorized the default role of the Data Integration system you can authorize the role by logging onto RAM using the primary account and then refresh the page.

- **Name**: A name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **Region**: The region you selected when purchasing Redis.
- **Redis instance ID**: You can go to the Redis console to view the Redis instance ID.
- **Redis access password**: The Redis Server access password. This field can be left blank, if there is no Redis access password.

The following figure is an example of adding a new Redis > ApsaraDB RDS type.

**Configurations:**

- **Type**: Currently, the selected data source type is Redis > On-premise Database with Public Network IP Address.
• Name: The name must start with a letter or underscore (_) and cannot exceed 60 characters in length. It can contain letters, numbers, and underscores (_).
• Description: A brief description of the data source that does not exceed 80 characters in length.
• Access address: The format is host:port.
• Add an access address: Add an access address in the format of host:port.
• Redis access password: The Redis service access password.

6. Click Test Connectivity
7. When the connectivity test is passed, click Complete.

This document explains how to configure the Redis Writer plug-in later. For more information, see Configure Redis Writer.

2.19 Configure HybridDB for MySQL data source

This topic describes detailed steps and related instructions for configuring the HybridDB for MySQL data source. The HybridDB for MySQL data source allows you to read/write data to HybridDB for MySQL.

You can configure synchronization tasks in Wizard mode and Script mode.

Note:
If the HybridDB for MySQL is in a VPC environment, you need to note the following issues:

• On-premise MySQL data source
  - Test connectivity is not supported, but supports synchronization task configuration. You can click OK when creating the data source.
  - You must use a custom scheduled resource group to run the corresponding synchronization tasks, make sure the custom resource group can connect to the on-premise database. For more information, see Data sync when one-end of the data source network is disconnected and Data synchronization when both-end of the data source network is disconnected.

• For the HybridDB of MySQL data sources created with an instance ID, you do not need to select a network environment, and the system automatically
determines the network environment based on the information you entered for the HybridDB for MySQL instance.

Procedure

1. Log on to the DataWorks console page using the administrator (primary account). Click Enter Workspace in the Actions column of the relevant project in the Project List.

2. Move mouse to the icon of DataWorks in the upper left corner, select Data Integration.

3. Click New Source in the data source page. Go to the supported data source pop-up window, as shown in the following figure.

![Add Data Source](image)

4. Select Alibaba Cloud Data Source (HybridDB) as the data source type in the new dialog box.
5. Configure the individual information items for HybridDB of the MySQL data source.

Configurations:

- **Type**: The currently selected data source type is the HybridDB for MySQL.
- **Name**: The name can contain letters, numbers, and underscores (_). It cannot start with a number or underscore (_).
- **Description**: A brief description of the data source that cannot exceed 80 characters in length.
- **Instance ID**: You can go to the HybridDB for MySQL console to view the related instance ID.
• Master account ID: You can view the related information in the security settings of the HybridDB for MySQL console.

![Security Settings](image)

- Login Account: [Redacted] (You have passed identity verification)
- Account ID: [Redacted]
- Registration Time: May 21, 2018 6:03:00 PM

• Username and password: The user name and password used for database connection.

6. Click Test Connectivity.
7. When the test connection is completed, click Complete.

Note:
You need to add a whitelist before connecting. For more information, see Add whitelist document.

The description of test connectivity

- The connectivity test is available in the classic network configuration.
- The private network can be added successfully in the form of adding an instance ID, and providing related reverse proxy function.

Next step

The next topic describes how to configure the HybridDB for MySQL Writer plug-in. For more information, see Configure HybridDB for MySQL Reader and Configure HybridDB for MySQL Writer.

2.20 Configure HybridDB for PostgreSQL data source

This topic describes the HybridDB and PostgreSQL data source. The HybridDB for PostgreSQL data source allows you to read/write data to HybridDB for PostgreSQL.
This topic introduces the detailed steps and related instructions for configuring the HybridDB for PostgreSQL data source.

You can configure synchronization tasks in **Wizard mode configuration** and **Script mode configuration**.

---

**Note:**

If the HybridDB for PostgreSQL is in a VPC environment, you need to note the following issues.

- **On-premise PostgreSQL data source**
  - On-premise PostgreSQL data source test connectivity is not supported, but supports synchronization task configuration. You can click confirm when creating the data source.
  - You must use a custom scheduled resource group to run the corresponding synchronization tasks, make sure the custom Resource Group can connect to the on-premise database. For more information, see #unique_13 and #unique_14.

- **HybridDB for PostgreSQL data sources created with an instance ID.**
  
  You do not need to select a network environment, the system automatically selects the network based on the HybridDB instance for PostgreSQL information.

---

**Procedure**

1. **Log on to the DataWorks console page as an administrator (primary account).** Click Enter Workspace in the Actions column of the relevant project in the Project List.
2. **Move mouse to the icon of DataWorks in the upper left corner, select Data Integration.**
3. Click New Source in the data source page of the supported data source pop-up window, as shown in the following figure.

4. Select HybridDB for PostgreSQL as the data source type in the new dialog box.
5. Configure the individual information items for the HybridDB for PostgreSQL data source. Consider a data source for the New HybridDB for PostgreSQL > Alibaba Cloud Database (HybridDB) type.

**Configurations:**

- **Type:** Alibaba Cloud HybridDB for MySQL
- **Name:** The data source name can contain letters, numbers, and underscores (_). It cannot start with a number or underscore(_).
- **Description:** A brief description of the data source that cannot exceed 80 characters in length.
- **Instance ID:** You can go to the HybridDB for PostgreSQL console to view the relevant instance ID.
· Master account ID: You can view the relevant information in the security settings of the HybridDB for PostgreSQL console.

### Security Settings

- Login Account: [User's account info]
- Account ID: [User's account ID]
- Registration Time: [Date and time]

Change Avatar

6. Click Test Connectivity.
7. When the test connection is completed, click OK.

**Note:**

You need to add a whitelist before connecting. For more information, see #unique_74 document.

The description of test connectivity

- The connectivity test is available in the classic network configuration.
- The private network can be added by adding the instance ID, and provides the related reverse proxy function.

Next step

The next topic describes how to configure the PostgreSQL Writer plug-in. For more information, see Configure AnalyticDB for PostgreSQL Reader and Configure HybridDB for PostgreSQL Writer.

### 2.21 Configure a POLARDB data source

A POLARDB data source is a relational database, which allows you to read data from and write data into it. This section describes how to configure a POLARDB data source.

You can configure synchronization tasks in wizard mode or script mode. For more information, see Wizard mode configuration and Script mode configuration.
Note:

Currently, POLARDB data sources do not support custom resource groups. Use the default resource group. If you need to use custom resource groups, add a MySQL data source and set the data source type to Public IP Address Unavailable. For more information, see *Configure MySQL data source*.

If your POLARDB data source is located in a VPC, pay attention to the following:

- For a user-created POLARDB data source
  - Connectivity testing is not supported, but you can still configure synchronization tasks. You can ignore the Test Connectivity button, and click Complete when you create the data source.
  - You must use a custom resource group to run the corresponding synchronization tasks. Make sure that the custom resource group can connect to the database that you have created. For more information, see *Data Integration when one side of the data source is disconnected* and *Data sync when both ends of the data source network is disconnected*.

- For a POLARDB data source created with the ID of a POLARDB instance
  You do not need to select the network environment, because the system automatically determines the network environment based on the information you enter for the POLARDB instance.

Procedure

1. Log on to the *DataWorks console* as an administrator (the primary account). In the Workspaces area of the Overview page, click Data Analytics in the Actions column of a workspace.

2. Move your pointer over the DataWorks icon in the upper-left corner, and select Data Integration.
3. Click Add Data Source on the Data Source page. A dialog box appears, listing the supported data source types, as shown in the following figure.

4. In the Add Data Source dialog box, select POLARDB as the data source type.
5. Set the parameters required for creating a POLARDB data source.

The parameters are described as follows:

- **Data Source Type**: The data source type. In this case, POLARDB is selected.
- **Data Source Name**: The data source name. The value must contain letters, digits, and underscores (_), but it must not start with a digit or underscore (_).
- **Description**: A brief description of the data source. The value can contain a maximum of 80 characters in length.
- **Cluster ID**: You can view the cluster ID in the POLARDB console.
- **Polardb instance main account ID**: You can view the primary account ID of the POLARDB instance on the Security Settings page of the POLARDB console.
- Database Name: The name of the database created in the POLARDB instance.
- Username and Password: The username and password used to connect to the database.

6. Click Test Connectivity.
7. After the connectivity test is passed, click Complete.

Note:
The connectivity test can be passed only after the data source is added to the whitelist. For more information, see Add a whitelist.

Connectivity test description
- The connectivity test is available in a classic network environment.
- In a VPC, you can add the data source successfully by adding the instance ID. In this case, the reverse proxy feature is provided to ensure that the data source can be connected.

Next step
Now you have learned how to configure the POLARDB data source. For more information about how to configure the POLARDB reader and writer plug-ins, see Configure POLARDB Reader and Configure POLARDB Writer.