Enable Relational Connections

Contents:

The Trifacta® platform can be configured to access data stored in relational database sources over JDBC protocol. When this connection method is used, individual database tables and views can be imported as datasets.

Supported Relational Databases

The Trifacta platform can natively connect to these relational database platforms. Natively supported versions are the following:

- Oracle 12.1.0.2
- SQL Server 12.0.4
- PostgreSQL 9.3.10
- Teradata 14.10+

> **NOTE:** To enable Teradata connections, you must download and install Teradata drivers first. For more information, see *Enable Teradata Connections*.

Additional relational connections can be enabled and configured for the platform. For more information, see *Connection Types*.

Ports

For any relational source to which you are connecting, the Trifacta node must be able to access it through the specified host and port value.

Please contact your database administrator for the host and port information.

Enable

This feature is enabled automatically.

> **NOTE:** Disabling this feature hides existing relational connections.

Disable relational publishing

By default, relational connections are read/write, which means that users can create connections that enable writing back to source databases.

- When this feature is enabled, writeback is enabled for all natively supported relational connection types. See *Connection Types*.
- Depending on the connection type, the Trifacta platform writes its data to different field types in the target database. For more information, see *Type Conversions*.

- Some limitations apply to relational writeback. See Limitations below.

As needed, you can disable this feature.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `false`:

```
"webapp.connectivity.relationalWriteback.enabled": true,
```

3. Save changes and restart the platform.

Publishing through relational connections is disabled.

Limitations

> **NOTE:** Unless otherwise noted, authentication to a relational connection requires basic authentication (username/password) credentials.

- You cannot swap relational sources if they are from databases provided by different vendors. See *Flow View Page*.
- There are some differences in behavior between reading tables and views. See *Using Databases*.

Limitations on relational publishing:

> **When the relational publishing feature is enabled, it is automatically enabled for all platform-native connection types. You cannot disable relational publishing for Oracle, SQL Server, PostgreSQL, or Teradata connection types. Before you enable, please verify that all user accounts accessing databases of these types have appropriate permissions.**

> **NOTE:** Writing back to the database utilizes the same user credentials and therefore permissions as reading from it. Please verify that the users who are creating read/write relational connections have appropriate access.

- You cannot ad-hoc publish to a relational target. Relational publishing is only supported through the Run Job page.
- You write to multiple relational outputs from the same job only if they are from the same vendor.
  - For example, if you have two SQL Server connections A and B, you can write one set of results to A and another set of results to B for the same job.
  - If A and B are from different database vendors, you cannot write to them from the same job.

Execution at scale

Jobs for large-scale relational sources can be executed on the Spark running environment. After the data source has been imported and wrangled, no additional configuration is required to execute at scale.

NOTE: End-to-end performance is likely to be impacted by:

- streaming data volumes over 1 TB from the source,
- streaming from multiple concurrent sources,
- overall network bandwidth.

When the job is completed, any temporary files are automatically removed from HDFS.

For more information, see *Run Job Page*.

Password Encryption Key File

Relational database passwords are encrypted using key files:

- **Passwords in transit:** The platform uses a proprietary encryption key that is invoked each time a relational password is shared among platform services.
- **Passwords at rest:** For creating connections to your relational sources, you must create and reference your own encryption key file. This encryption key is accessing your relational connections from the web application. For more information, see *Create Encryption Key File*.