What would you do if you knew?™

Teradata QueryGrid

User Guide

Release 2.0

B035-1205-100K

September 2016

Table of Contents

Preface

Purpose

This book describes Teradata® QueryGrid™ for seamless, high-performing data access, processing, and movement across systems in heterogeneous analytical environments.

Use this book with the other books in the SQL book set.

Audience

This book is intended for database administrators and other technical personnel who use Teradata QueryGrid.

Revision History

Release	Description
Release 2.0 September 2016	Initial release.

Supported Releases

For information on Teradata QueryGrid supported releases, see Knowledge Article KAP314E23E, accessed through https://tays.teradata.com.

Additional Information

Related Links

URL	Description
https://tays.teradata.com	Use Teradata At Your Service to access Orange Books, technical alerts, and knowledge repositories, view and join forums, and download software packages.
www.teradata.com	External site for product, service, resource, support, and other customer information.

Related Documentation

Documents are located at http://www.info.teradata.com.

Title	Publication ID
Teradata QueryGrid Installation, Configuration, and Upgrade Guide This guide explains how to install, configure, and upgrade Teradata QueryGrid.	B035-5990

Customer Education

Teradata Customer Education delivers training for your global workforce, including scheduled public courses, customized on-site training, and web-based training. For information about the classes, schedules, and the Teradata Certification Program, go to www.teradata.com/TEN/.

Software Downloads

URL	Description
Teradata Developer Exchange	Teradata Developer Exchange provides articles on using Teradata products, technical discussion forums, and code downloads.

Product Safety Information

This document may contain information addressing product safety practices related to data or property damage, identified by the word *Notice*. A notice indicates a situation which, if not avoided, could result in damage to property, such as equipment or data, but not related to personal injury.

Example

Notice:
Improper use of the Reconfiguration utility can result in data loss.

CHAPTER 1

Overview

Teradata QueryGrid Description

Teradata QueryGrid provides seamless, high-performing data access, processing, and movement across systems in heterogeneous analytical environments. It comprises the following main structural components:

- Teradata QueryGrid manager enables definition, administration, and monitoring of your Teradata QueryGrid ecosystem.

  The **QueryGrid** portlet in Teradata Viewpoint provides access to the manager functionality. You use the portlet to install and configure the Teradata QueryGrid components and monitor the health of those components.
- Teradata QueryGrid connector provides for query processing across data sources, such as a Teradata Database system and a Hadoop system on which the Teradata Distribution of Presto is installed.

  Connectors translate the request language of queries and transform data so that it can be exchanged between different types of systems. Connectors control the execution of queries on target systems and return the results to initiating systems. Connectors run on one or more nodes in a system, and are specific to the system type, such as Teradata or Presto-configured Hadoop. Optional properties allow for configuration refinement of each connector type.
- Teradata QueryGrid link specifies which connectors can communicate with each other and defines rules of data transfer.

  Each link specifies a pair of connectors: an initiating connector, which is the point from which a query originates, and a target connector, which is the destination point of the query. Likewise, links define initiating and target networks. Each link also specifies a communications policy that defines rules for transferring data between connectors. Optional properties allow for configuration refinement of the initiating or target connector. After Teradata QueryGrid installation, you can use links to simplify configuration of foreign server definitions in preparation for building queries.

Connectors and links are associated with a fabric. A fabric represents a set of interconnected nodes, all of which run compatible versions of Teradata QueryGrid software over the same port. Each fabric organizes Teradata QueryGrid components into a discrete environment and enables communication between paired data sources of the same or differing type within that environment.

Fabrics, connectors, and links are considered configurations, each of which can have three co-existing versions:

- Active
- Pending
- Previous

This feature allows you to test new connectors and fabrics on production deployments without affecting production workloads, and allows time to push changes to all systems before being utilized by production queries.

Data Dictionary Views for Teradata QueryGrid

The following Data Dictionary views are available for Teradata QueryGrid:

- DBC.QryLogV (TotalServerByteCount column only)
- DBC.QryLogStepsV (ServerByteCount column only)
- DBC.ServerV[X]
- DBC.ServerInfoV[X]
- DBC.TblSrvV[X]
- DBC.TblSrvInfoV[X]

For more information, see *Data Dictionary*, B035-1092.

Teradata QueryGrid Considerations

The following list describes the considerations to be aware of when using this release of Teradata QueryGrid.

- You can continue to use prior-generation Teradata QueryGrid connectors independently of this release. However, those connectors cannot be installed, configured, or monitored using the **QueryGrid** portlet in Viewpoint.
- This release is compatible with the SQL data manipulation language used for prior-generation Teradata QueryGrid connectors.
- Previous foreign server definitions continue to work with the older connectors.
- If you want to use a Teradata QueryGrid connector, you must create a new foreign server definition. If you do not want to change your existing SQL, you can use the old foreign server name after performing the following steps:

  1. Stop all queries against the old foreign server.
  2. Drop the old foreign server.
  3. Create a new foreign server using the old foreign server name. Configure the new foreign server with the name value pairs from the current release of Teradata QueryGrid. For details about name value pairs, see Connector and Link Properties.

Teradata QueryGrid Limitations

The following list describes Teradata QueryGrid limitations for this release:

- Supports data transfer over the wire in INDIC format only. Indicator format is an efficient client data format that represents fixed and variable length data type values and includes indicator bits for each column to indicate null values.
- Does not support transaction semantics between systems.

  After data has been exported and committed to a remote system, any subsequent errors or aborts on the local system do not roll back the remote request.
- Does not support the following Teradata data types: period, geospatial, XML, JSON, BLOB, CLOB, and UDT.
- Does not support the following Presto data types: JSON, ARRAY, time, map, and row.

ANSI Compliance

The syntax used for the connector is a Teradata extension to the ANSI SQL:2011 standard.

CHAPTER 2

Teradata QueryGrid Privileges and Security

Teradata QueryGrid Authentication, Integrity, and Encryption

All communication between the Teradata QueryGrid manager instances and the Teradata QueryGrid attached nodes uses HTTPS over port 9444. When the first manager first comes online, it generates a Root Certificate Authority that is used to issue all the other managers a server certificate. This trusted CA certificate is given to each of the nodes at the time of installation, allowing them to verify that they are communicating with a trusted Teradata QueryGrid manager.

When nodes first register with a manager, they specify an auto-generated UUID that acts as their username, a time-limited access token that is associated with system they want to join, and a secret password to use to authenticate going forward. The secret password is salted using a cryptographically secure pseudo random number generator and is one-way encrypted. If the system access token is valid, the node is allowed to join the system.

The Teradata QueryGrid fabric supports several levels of security options that you can configure using a communication policy:

- The least secure level uses IP-based authentication, no integrity check, and no encryption. This is recommended only for highly trusted environments.
- The second level of security, IP-based authentication, checksum integrity check, and no encryption, is also recommended for use only in trusted environments.
- The next level is key-based authentication, secure integrity check, and no encryption.
- For tighter security, we recommend that you use key-based authentication, secure integrity check, and encrypted credentials.
- For even tighter security, we recommend that you use key-based authentication, secure integrity check, and the encryption of all data. You can choose to use AES-CRC32, AES-GCM (default), AES-SHA256, or AES-SHA512 as your encryption and integrity algorithm.

For information about setting the security level, see Adding a Communication Policy.

Security Guidelines

You can maintain security for Teradata QueryGrid by:

- Changing Teradata QueryGrid manager service account default passwords on install.
- Setting appropriate permissions in Viewpoint to limit the users that have access to edit the Teradata QueryGrid configuration.
- Only creating links using initiator data sources whose environments are secure.
- Using communication policies security options appropriate to the environment.

- Setting appropriate privileges to those who create and manage foreign servers.
- Setting appropriate privileges to authorization objects, if used. (An authorization object stores a mapping of the local user and password for the remote server.)
- Setting up foreign server objects that match the appropriate access to database and tables needed by Teradata Database users.
- Setting appropriate privileges to foreign server objects to Teradata Database users.

Additional Guidelines for Security

You may want to consider the following security guidelines:

- You should not grant EXECUTE FUNCTION privileges on the functions in SYSLIB or TD_SYSFNLIB to users performing queries on the foreign server.
- GRANT CREATE and DROP SERVER privileges only to a trusted database administrator who administers the server setup.
- The trusted database administrator can then grant SELECT or INSERT privilege on the server objects to a subset of users.
- The trusted database administrator can set up authentication using one of the methods described in Kerberos Authentication Security.
- If a Kerberos external security system is used by the Hadoop cluster, the user specified in an authorization object must exist in the external security system.
- The user may or may not belong to any group on the Hadoop cluster.

Security and User Sessions

Each type of Teradata QueryGrid connector uses different mechanisms for authenticating the local user on the remote system.

Teradata and User Sessions

Teradata has a mechanism in the foreign server definition called an authorization object, which contains the authenticating username and password for the remote system. This works for either a one-to-one user mapping or a many-to-one user mapping (used for trusted users or service accounts). See *SQL Data Definition Language - Detailed Topics*.

Teradata also has a user mapping mechanism associated with the foreign server definition that permits the mapping of user names. This permits users to be mapped for data centers that do not use a common user identification across all systems. You can configure the user mapping table in the **QueryGrid** portlet; see User Mappings View.

Teradata supports LDAP. This means that queries with a Teradata target can be submitted using the LDAP security mechanism to authenticate; for example, the Presto-to-Teradata connection supports LDAP.

Teradata provides a user configurable mechanism for authenticating to Kerberos. The user and password is used to obtain a ticket from the KDC when attempting to connect to a remote Kerberized Teradata or Presto.

Presto and User Sessions

Presto supports Kerberos authentication.

Presto user credentials are stored in the Username and Password properties in the connector or link settings in Teradata QueryGrid. For information about setting Username and Password in the **QueryGrid** portlet, see Connector and Link Properties and Adding a Connector.

Session Settings

Session settings are passed from the local Teradata Database to a remote Teradata Database:

- ANSI or BTET mode - affects transaction semantics
- QueryBand - affects the remote query's workload management

In the Trusted Session mode for a remote Teradata, the PROXYUSER and PROXYROLE QueryBand settings are used.

Note:
PROXYROLE requires Teradata Database 16.0 or higher.

Presto does not support session settings.

Kerberos Authentication Security

You can set up Teradata QueryGrid to use Kerberos authentication with the Presto connector. The Presto connector uses two forms of authentication with Kerberos:

Authorization Object and Kerberos

For a Teradata-to-Presto connection, create an authorization object and use it in the USING clause of the foreign server. Teradata passes the credentials in an authorization object to the Presto connector, then the Presto connector authenticates the user against Kerberos before sending the query to the data source.

Kerberos Keytab

Presto can be configured to enable Kerberos authentication over HTTPS for clients.

Authorization Object and Kerberos

Authorization Object and Kerberos

You can create an authorization object, which stores the credentials for a user in Kerberos in encrypted form. You can set up the Presto connector to use authorization objects.

If you need one-to-one mapping between a Teradata Database user and a remote user, then you must have corresponding accounts in Teradata Database and the security system. When that user creates the authorization using AS INVOKER TRUSTED, the authorization is stored by default on the user database. The credentials for the security system do not need to be revealed to another person and the authorization object is accessible only to users with privilege to that database.

You can use many-to-one mapping between multiple Teradata Database users and one user on the remote system to simplify administration. Only the creator of the authorization need know the credentials for the user on the remote security system. When the authorization is created using AS DEFINER TRUSTED, the authorization is stored by default in the TD_SERVER_DB database, which makes the authorization available globally.

Where the Foreign Server Looks for the Authorization Object

When a foreign server is configured with the INVOKER keyword and no value is specified for the database name (dbname) the Presto connector automatically looks for the authorization in the user database of the session user.

When a foreign server is configured with the DEFINER keyword and no value is specified for the database name (dbname) the Teradata connector automatically looks for the authorization in the TD_SERVER_DB database.

For information on creating an authorization, see CREATE AUTHORIZATION and REPLACE AUTHORIZATION.

Example: Kerberos Using INVOKER

This example creates the `remote_presto1` authorization object in the creator's user database. If the creator is `td_user` then `td_user.remote_presto1` is the fully qualified object name.

```
create authorization remote_presto1 as invoker trusted
      user 'kerberos_user' password 'kerberos_pass';
```

This example creates a foreign server object that uses the `remote_presto1` authorization object.

```
create foreign server Presto_1
external security invoker trusted td_user.remote_presto1
using
 link('td1_presto1')
 version('active')
DO IMPORT WITH TD_SYSFNLIB.QGInitiatorImport,
DO EXPORT WITH TD_SYSFNLIB.QGInitiatorExport;
```

The **link** value of `td1_presto1` was created in the **QueryGrid** portlet. The link defines initiating and target networks, specifies a communications policy that defines rules for transferring data between connectors, and other properties.

Example: Kerberos Using DEFINER

This example creates the `remote_presto1` authorization object in the TD_SERVER_DB database.

```
create authorization td_server_db.remote_presto1 as definer trusted
      user 'kerberos_proxy_user' password 'kerberos_proxy_pass';
```

This example creates a foreign server object that uses the remote_presto1 authorization object.

```
create foreign server Presto_1
external security definer trusted TD_SERVER_DB.remote_presto1
using
 link('td1_presto1')
 version('active')
DO IMPORT WITH TD_SYSFNLIB.QGInitiatorImport,
DO EXPORT WITH TD_SYSFNLIB.QGInitiatorExport;
```

The **link** value of `td1_presto1` was created in the **QueryGrid** portlet. The link defines initiating and target networks, specifies a communications policy that defines rules for transferring data between connectors, and other properties.

Setup Process for Kerberos Authentication

Presto can be configured to enable Kerberos authentication over HTTPS for clients. The driver node on the remote server establishes Kerberos authentication using the remote keytab file. The Presto connector is configured with the location of the needed file.

For detailed information about setting up the Presto Kerberos configuration, see *Teradata Presto Documentation* available from http://www.info.teradata.com: **Teradata for Hadoop** > **Teradata Distribution of Presto** > **Teradata Distribution of Presto**

Kerberos Maintenance

You must update the configuration of Teradata QueryGrid if the name or location of the default Kerberos realm or the location of the host for your KDC (Key Distribution Center) or administration server changes.

CHAPTER 3

Administration and Utilities for Teradata QueryGrid

Monitoring User Queries Between Local and Remote Servers

Using Viewpoint to Monitor Queries

Use the Viewpoint **Query Monitor** portlet to monitor the queries running on the source and target system. Viewpoint **Completed Queries**, **My Queries**, and **Query Groups** portlets also display Teradata QueryGrid operation details.

Query performance data includes metrics about each active query:

- CPU
- CPU Skew
- Rows Transferred
- Bytes Transferred
- Data Skew

These messages include local and remote session details, query text, errors, and durations for all query phases.

The data is captured from all nodes participating in the query for a particular query. The captured data is periodically summarized. The summarization frequency is configurable in Viewpoint, but defaults to 30 seconds.

For information, see *Teradata Viewpoint User Guide*, B035-2206.

Logging and Error Handling

Using the QueryGrid Portlet to View Issues

The **Issues** view in the Teradata Viewpoint **QueryGrid** portlet allows you to review possible problems with Teradata QueryGrid. Some of these issues may require that you take action. The following types of issues are reported:

- Teradata QueryGrid component has gone offline.
- Teradata QueryGrid manager has failed.
- Teradata QueryGrid manager backup failed.
- Teradata QueryGrid manager is running out of disk space.
- One or more Teradata QueryGrid managers are running different versions of the Teradata QueryGrid manager package.
- One or more nodes on the system are missing the connector package.

- System time is incorrect on a node.

Each condition has an associated severity, either critical or warning.

To view issues history from the **QueryGrid** portlet, see Viewing Issues History.

Teradata QueryGrid Portlet Logging Settings

You can set the log data retention and query data retention times in the **QueryGrid** portlet. See Editing Manager Settings.

Utilities

Resetting a Service Account Password

Use a command line tool `reset-password.sh` to reset a service account password on the Teradata QueryGrid manager system.

1. From the command line of the Teradata QueryGrid manager system, run: `/opt/teradata/tdqgm/bin/reset-password.sh`
   The user must have sudo privilege or be a member of the tdqgm group to run this command.

Adding a QueryGrid Manager to a Cluster

Use the command line utility `join-cluster.sh` to add a Teradata QueryGrid manager to a cluster.

Note:
The Teradata QueryGrid manager package (file name `tdqg-manager-`*`version`*`.rpm`) must be installed on the system to join a cluster.

1. To cluster Teradata QueryGrid manager instances:
   a) Run the `join-cluster` script:

   ```
   /opt/teradata/tdqgm/bin/join-cluster.sh
   ```

   b) Enter the hostname of a Teradata QueryGrid manager instance in the cluster.
   c) Specify the login credentials for a user granted read permission to `/etc/opt/teradata/tdqgm` on that instance.
   d) Enter the Data Center for this instance.
      If the applicable Data Center is not listed, press **Ctrl+C** to cancel, then use the **QueryGrid** portlet in Teradata Viewpoint to add the Data Center.
   e) Enter `y` at the prompt to continue with the clustering process.

Resetting QueryGrid Manager to Default Settings

Use the command line utility `reset.sh` to reset a Teradata QueryGrid manager (manager) back to its default settings.

The options for `reset.sh` are:

```
./reset.sh --help

usage: reset [-f] [-h]

Resets this QueryGrid Manager back to its factory settings.

options:
-f,--force   Force the reset without prompting for confirmation.
-h,--help    Print this message
```

1. Log on to the system where the Teradata QueryGrid manager is running.
2. From `/opt/teradata/tdqgm/bin`, run: `./reset.sh -f` and answer `y` when it prompts you for confirmation.
   Result: The manager is reported as offline.

Removing QueryGrid Manager from a Cluster

Prerequisite

Use the command line utility `reset.sh` to reset the Teradata QueryGrid manager (manager) back to its default settings. This sets the status of the manager to offline.

After resetting the manager, you can remove it from a cluster using the **QueryGrid** portlet in Teradata Viewpoint.

1. Log on to the system where the Teradata QueryGrid manager is running.
2. From `/opt/teradata/tdqgm/bin`, run: `./reset.sh -f` and answer `y` when it prompts you for confirmation.
   Result: The manager is reported as offline.
3. In Teradata Viewpoint, in the **QueryGrid** portlet, from the **Managers** view, in the **Managers** tab, select **Delete** for the manager you want to delete.

   Note:
   You can only delete managers with a status of offline.

Restarting a Fabric

Use the command line utility `fabric-restart.sh` to restart a fabric.

1. Log on to the system where the Teradata QueryGrid manager is running.
2. From `/opt/teradata/tdqgm/bin`, run: `./fabric-restart.sh` and answer the prompts.

```
Fabrics
-------
1. Production (ACTIVE)
2. Production (PENDING)

Select a fabric [1-2]: 2
```

```
Systems
-------
1. ALL SYSTEMS
2. HdpProd
3. TD1

Select a system [1-3]: 1

Are you sure you want to restart Production (PENDING) on ALL SYSTEMS? [y/
n]: y
```

Removing Old Software Versions from Teradata QueryGrid Manager

Use the command line utility `cleanup-software.sh` to remove old software versions from Teradata QueryGrid manager. The utility cleans up fabric, node, and connector software that is unused for the specified period of time.

The options for `cleanup-software.sh` are:

```
./cleanup-software.sh --help
usage: cleanup-software [-d <days>] [-h] [-m <host>] [-n]

Cleanup older software that is no longer being used.

options:
 -d,--days <days>      The number of days old the software should be
                       before considering it for removal. Defaults to 7.
 -h,--help             Print this message
 -m,--manager <host>   The hostname of the QueryGrid manager to connect
                       to. Defaults to localhost.
 -n,--no-ssl           Use HTTP instead of HTTPS to connect to the manager.
```

1. Log on to the system where the Teradata QueryGrid manager is running.
2. From `/opt/teradata/tdqgm/bin`, run: `./cleanup-software.sh`

```
Removing unused software older than 2016-08-25T10:26:41.356-0700.
Deleted tdqg-fabric-02.00.00.00_SNAPSHOT.tar.gz.
Deleted tdqg-presto-connector-02.00.00.00_SNAPSHOT.tar.gz.
Deleted tdqg-node-02.00.00.00_SNAPSHOT.tar.gz.
Deleted tdqg-teradata-connector-02.00.00.00_SNAPSHOT.tar.gz.
[…]
```

Archive and Restore

Teradata QueryGrid manager provides commands for backing up and restoring the Teradata QueryGrid configuration. Because the manager provides a clustering capability that involves replication of configuration to all manager instances, it is typically unnecessary to restore from backup after a failure; as long as one manager instance is still online, the configuration is preserved. However, there are certain circumstances that may require a restore from backup:

- Recovery from a catastrophic failure where all manager instances were lost
- Cloning an existing configuration into a new cluster

When restoring from a backup, it is important to note that the manager instance being restored becomes a new manager cluster. To add existing managers to the new manager cluster, run `/opt/teradata/tdqgm/bin/join-cluster.sh` on each manager instance, one at a time. Wait for each instance to come online before starting the next join cluster.

Each manager instance will automatically generate a backup file once a day located in `/var/opt/teradata/tdqgm/backups`. Up to 7 days of backups are maintained.

Backup

The backup command is located at `/opt/teradata/tdqgm/bin/backup.sh`. By default , backups are placed in the `/var/opt/teradata/tdqgm/backups` directory and are named `tdqgm-backup-YYYY-MM-DD.zip` where YYYY-MM-DD is the current date. The backup file includes all the configuration details and software packages that were uploaded to the Teradata QueryGrid manager.

Usage

To get help for the backup command, use the --help option:

```
/opt/teradata/tdqgm/bin/backup.sh --help
```

Result:

```
usage: backup [-f <file>] [-h] [-x]
Creates a backup file that enables the QueryGrid manager configuration to
be restored in the event of a catastrophic failure.

options:
 -f,--file <file>        The path to the backup file to create.
 -h,--help               Print this message.
 -x,--exclude-software   Flags if software files should be excluded from the
backup.
```

Example: Backing Up Teradata QueryGrid

To perform the back up, run:

```
/opt/teradata/tdqgm/bin/backup.sh
```

Result:

```
Starting backup...

Backup completed:
  /var/opt/teradata/tdqgm/backups/tdqgm-backup-2016-06-16.zip
```

Restore

The restore command is located at `/opt/teradata/tdqgm/bin/restore.sh`. It requires the name of the backup file to restore as an argument.

Usage

To get help for the backup command, use the --help option:

```
/opt/teradata/tdqgm/bin/restore.sh --help
```

Result:

```
usage: restore [options] <backup-file>

Restores the QueryGrid manager configuration from a backup file in the event of a
catastrophic failure.

options:

 -a,--public-address <address>   The address other managers and nodes use
                                 to access this manager instance.
 -d,--data-center <name>         The name of the data center to join.
 -f,--force                      Force the restore to start without requiring
confirmation.
 -h,--help                       Print this message
```

Example: Restoring Teradata QueryGrid

To restore the system, run:

```
/opt/teradata/tdqgm/bin/restore.sh <backup_file>
Enter public address for this manager instance [<public_address>]:
```

where `<backup_file>` is the path name of the backup ZIP file, similar to `~/tdqgm-backup-2016-06-16.zip` and `<public_address>` is the hostname or IP address of the Teradata QueryGrid manager system; for example: test.mydomain.com or 198.51.100.20.

Result (note, enter y at the prompt):

```
A restore from backup will remove all data previously saved on this manager
instance and
all local manager services will be restarted.  This manager instance will become a
standalone cluster.
Other managers can join this instance using the join-cluster command.

Do you wish to continue? [y/n]:   y

Stopping QueryGrid Manager...

Deleting previous state...

Restoring configuration data...

Data Centers
------------
1.  San Diego

2.  Las Vegas

3.  Atlanta

4.  Richmond

Select data center of local manager instance [1-4]: 3

Restoring tdqg-node-16.00.00.00.DEV-30.tar.gz...
Restoring tdqg-teradata-connector-1.0-224-SNAPSHOT.tar.gz...
Restoring tdqg-fabric-1.00.00.00.tar.gz...
Restoring tdqg-teradata-connector-1.0-12-SNAPSHOT.zip...

Starting QueryGrid Manager...
Restore completed successfully.
```

CHAPTER 4

Using the Teradata Connector With Teradata QueryGrid

Creating the Server Database

To use the Teradata QueryGrid connectors, the TD_SERVER_DB database must exist to hold server objects and their associated information. This database is created by running the Database Initialization Program (DIP). DIP is a series of executable SQL script files packaged with Teradata Database. Each DIP script creates one or more system users, databases, macros, tables, and views, for use by the Teradata Database and/or by users.

All of the DIP scripts that you need should have been executed during Teradata Database installation.

For information about using the DIP scripts, see *Utilities*, B035-1102.

About Teradata Database Archive and Restore

Archiving and restoring the Teradata QueryGrid configuration is discussed in Archive and Restore. The following discusses what needs to be backed up in the Teradata Database to preserve the Teradata QueryGrid objects that are stored in the Teradata Database.

Database TD_SERVER_DB stores all the database objects created using the Teradata QueryGrid connector. Note the following about archiving or restoring this database:

- Archive and restore TD_SERVER_DB as a user database. It is not archived and restored as part of the DBC database.
- You can archive and restore the entire database or individual objects in the database.
- Teradata archives the associated rows from DBC.ServerInfo and DBC.ServerTblOpInfo at the same time as it archives TD_SERVER_DB.
- The post-restore script validates server connectivity.

Note the following about copying foreign server objects:

- Users lose their privileges on a foreign server object after it is copied, so administrators must grant these privileges again.
- You can copy the entire TD_SERVER_DB database or individual objects in the database. Renaming is not allowed.

Teradata QueryGrid Table Operators and TD_SERVER_DB Archive and Restore for Installs and Upgrades

Teradata QueryGrid table operators reside in TD_SYSFNLIB and are installed by a DIP script when the Teradata Database is installed. TD_SERVER_DB is also created by DIP during installation.

The Teradata QueryGrid table operators are not deleted when the system is upgraded, so the DIP script does not need to be rerun for an upgrade. When upgrading, the TD_SERVER_DB database can be restored from backup without rerunning the DIP scripts.

Privileges Needed to Use Teradata QueryGrid

Privileges for Administrators

CREATE SERVER and DROP SERVER are object-level privileges that restrict who can use the CREATE FOREIGN SERVER and DROP FOREIGN SERVER SQL statements.

- CREATE SERVER can only be granted on the TD_SERVER_DB database as a whole.
- DROP SERVER can be granted on the TD_SERVER_DB database or on individual foreign server objects.
- The CREATE SERVER and DROP SERVER privileges are included if you grant ALL privileges on the TD_SERVER_DB database.

In addition to the CREATE SERVER and DROP SERVER privileges, administrators need the EXECUTE FUNCTION and SELECT privileges on the import and export table operators or on the TD_SYSFNLIB database that contains the table operators in order to create, drop, and modify foreign server objects.

The creator of a foreign server object implicitly receives the following privileges on the object:

- SHOW privilege WITH GRANT OPTION
- DROP SERVER privilege WITH GRANT OPTION
- SELECT privilege WITH GRANT OPTION
- If the foreign server object is capable of exporting data (that is, the CREATE FOREIGN SERVER statement includes the DO EXPORT WITH clause), the creator automatically receives the INSERT privilege WITH GRANT OPTION

CREATE AUTHORIZATION and DROP AUTHORIZATION privileges are required to work with authorization objects referenced by foreign server objects. DROP AUTHORIZATION is automatically granted to the creator of an authorization object.

Privileges for Users of the Foreign Server Object

- You must grant SELECT, INSERT, and SHOW privileges on the foreign server object to users who query the remote database.
- Granting the ALL privilege on a foreign server object implicitly grants other privileges that depend on the nature of the foreign server:
  - If the foreign server object can import data from the remote database (that is, the CREATE FOREIGN SERVER statement included a DO IMPORT WITH clause), granting the ALL privilege on the foreign server implicitly includes the SELECT, SHOW, and DROP privileges.
  - If the foreign server object can export data to the remote database (that is, the CREATE FOREIGN SERVER statement included a DO EXPORT WITH clause), granting the ALL privilege on the foreign server implicitly includes the INSERT, SHOW, and DROP privileges.

SQL Used With the Teradata QueryGrid Connector

The following describes the SQL syntax and options specific to the Teradata QueryGrid connector, and provides examples of their use.

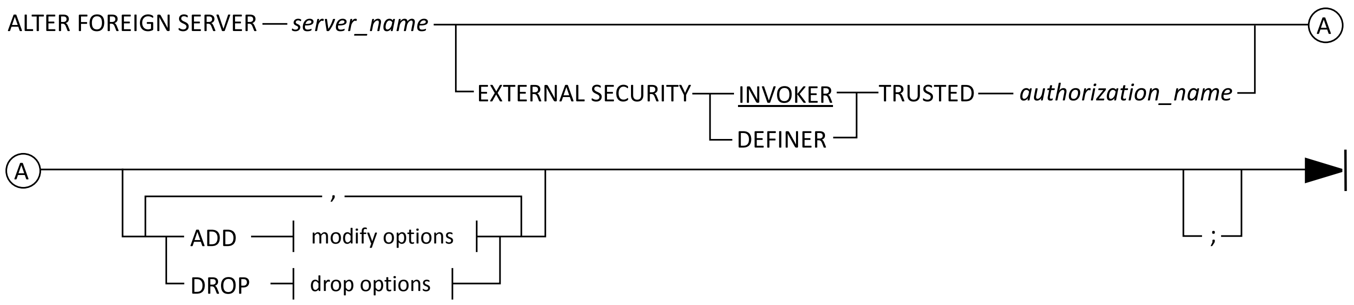
If you are a DBA, or a DBA has already granted you the privileges needed to create foreign servers, we recommend that you start by reading CREATE FOREIGN SERVER.

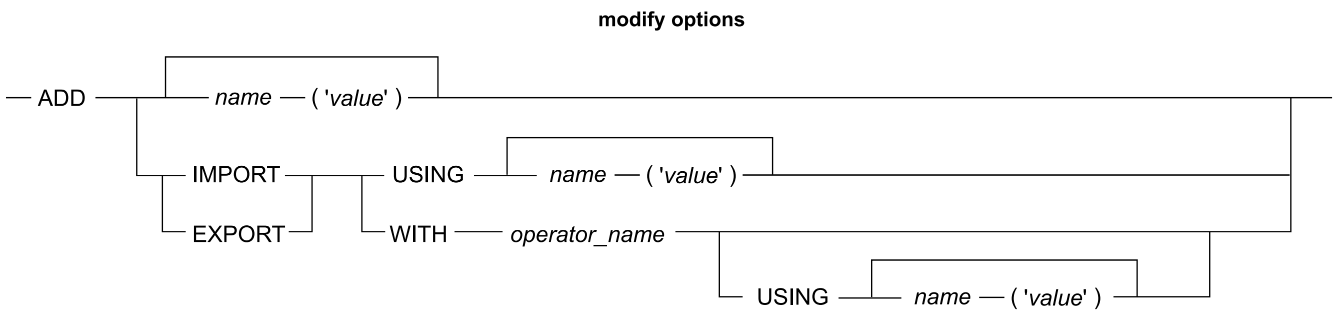
ALTER FOREIGN SERVER

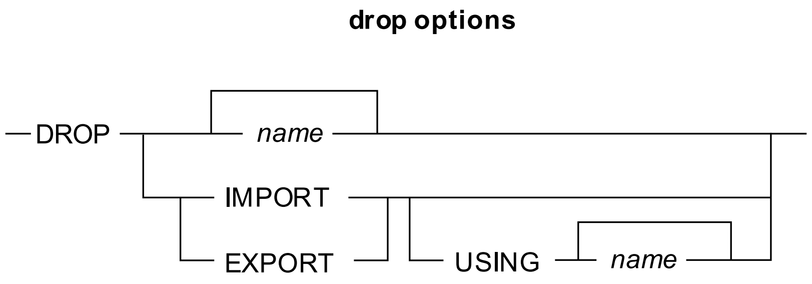
Purpose

Modifies the parameters of an existing server object.

Syntax







Syntax Elements

server_name

The name given to the foreign server object.

EXTERNAL SECURITY

Associates an authorization object with the foreign server. The authorization stores the encrypted credentials for a user as a database object. The Teradata connector passes the credentials in the authorization to the remote platform identified by the foreign server when the foreign server is accessed.

INVOKER
DEFINER

You can specify either a DEFINER or an INVOKER, but not both. If neither INVOKER nor DEFINER are specified then INVOKER is used by default.

INVOKER is a keyword that indicates that the associated authorization must be present in the user database at the time that the foreign server is accessed.

Note:
The user database is the database that was created for the user in the Teradata system when the user account was created.

DEFINER is a keyword that indicates that the associated authorization must be present in the database that contains the foreign server when the foreign server is accessed.

Note:
The DEFAULT keyword that can be used with DEFINER in CREATE AUTHORIZATION and REPLACE AUTHORIZATION statements is not needed in association with a foreign server.

If you use the DEFINER keyword, the authorization is global for all users who need to query the foreign server.

You must use either INVOKER TRUSTED or DEFINER TRUSTED if the remote platform uses an external security system (such as Kerberos, for example) for authentication.

TRUSTED

A keyword that indicates the associated authorization object was created as TRUSTED.

You must use the TRUSTED security type for the Teradata connector.

authorization_name

Specifies the name of the authorization object to be used when the foreign server is accessed.

Modify options

ADD

Use to:

- add or replace a global name value pair that is used to define the server object
- add an IMPORT or EXPORT table operator. If you want to replace a table operator that is already associated with the foreign server you must first drop the table operator before adding the new one.

***name*('*value*')**

The External Name Value pair or pairs (LINK and VERSION) that you want to add or modify.

The name value pair or pairs define the properties of the foreign server. The name value pairs are defined in the **QueryGrid** portlet. You can create separate named link configurations for unique sets of properties and you can create different versions (active, pending, previous) of the links. For more information, see Connector and Link Properties.

You can change the External Name Value pairs in ALTER FOREIGN SERVER. If you want to modify the properties of the named link, use the **QueryGrid** portlet.

Note that in the description of the name value pairs:

- The label "Server only" indicates that the name value pair must follow the syntax `ADD` *`name('value')`*.
- The label "Import only" indicates that the name value pair must be specified after the IMPORT keyword.
- The label "Export only" indicates that the name value pair must be specified after the EXPORT keyword.
- Unlabeled name value pairs may be specified in any part of the ADD syntax. If specified as `ADD` *`name('value')`*, the name value pair will be applied to the server as a whole.

IMPORT

Indicates that you are going to act on the operator that is used to import data into Teradata Database.

EXPORT

Indicates that you are going to act on the operator that is used to export data out of Teradata Database.

operator_name

The name of the table operator that you want to use.

For more information about the table operators used with the server object, see CREATE FOREIGN SERVER.

Drop options

DROP

- to drop a global name value pair that was used to define a server object. You need only specify the name to drop the pair.
- to drop an IMPORT or EXPORT table operator that was associated with a server definition. When you drop a table operator all related name value pairs are also dropped.
- to drop a local name value pair that was used with an IMPORT or EXPORT table operator. You need only specify the name to drop the pair.

name

When used alone, name is the name of the name value pair that you want to drop.

IMPORT

Indicates that you are going to act on the operator that is used to import data into Teradata Database.

EXPORT

Indicates that you are going to act on the operator that is used to export data out of Teradata Database

Required Privileges

You must have DROP SERVER privilege on the TD_SERVER_DB database or on the specified foreign server to modify the foreign server object. If you are modifying the table operators that are associated with the server, or adding a table operator, you must also have EXECUTE FUNCTION and SELECT privileges on the specified table operators.

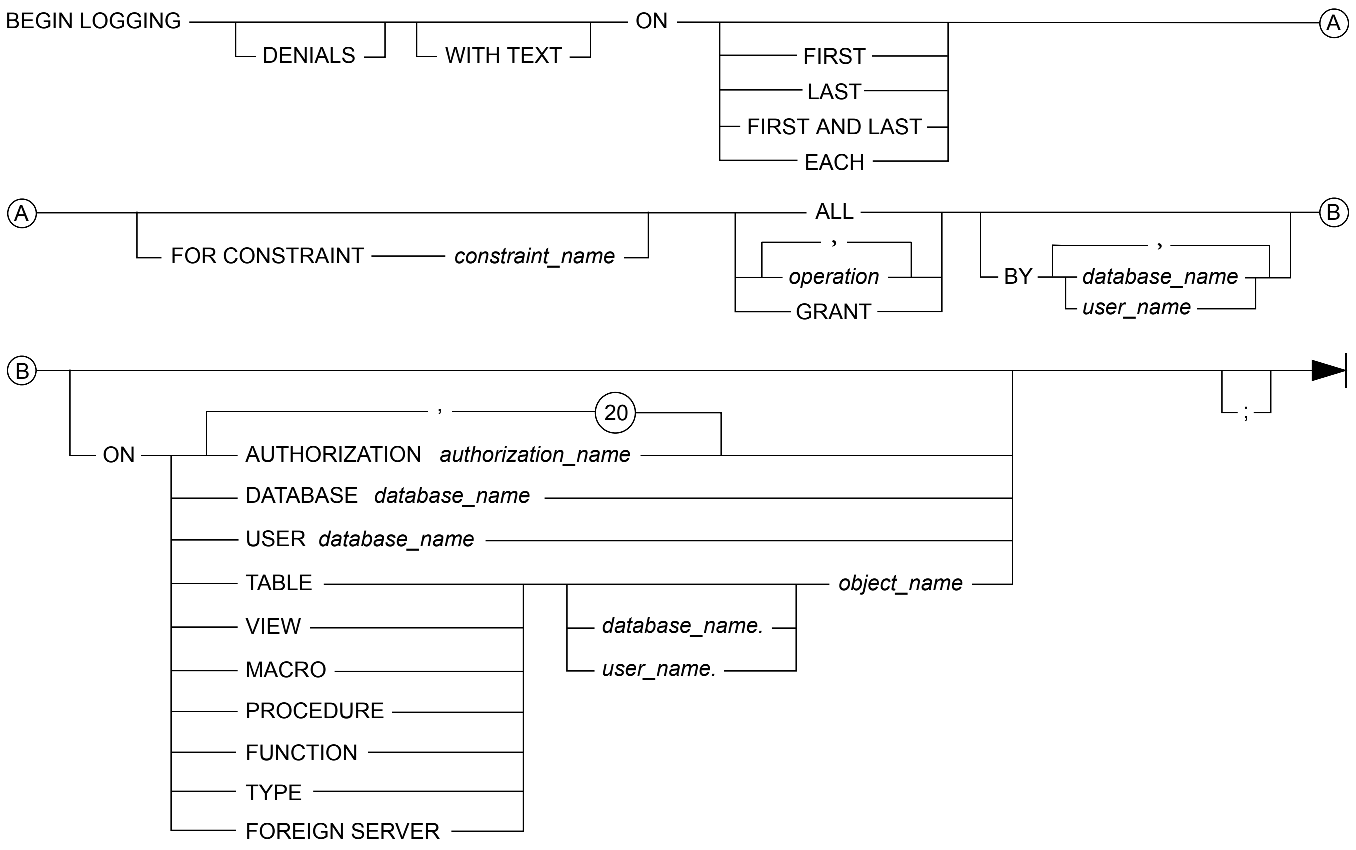
BEGIN LOGGING

Purpose

Starts the auditing of SQL requests that attempt to access data.

This topic describes only the portions of the BEGIN LOGGING syntax diagram that are specific to this Teradata QueryGrid connector. For information about the other syntax that you can use with BEGIN LOGGING, see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

Syntax



Syntax Element

ON FOREIGN SERVER *object_name*

Indicates that the database object for which access is to be logged is a foreign server.

You must specify an object name, which is the name of the foreign server. You can optionally specify the name of the containing database, which must be TD_SERVER_DB. You cannot use a *user_name* with FOREIGN SERVER.

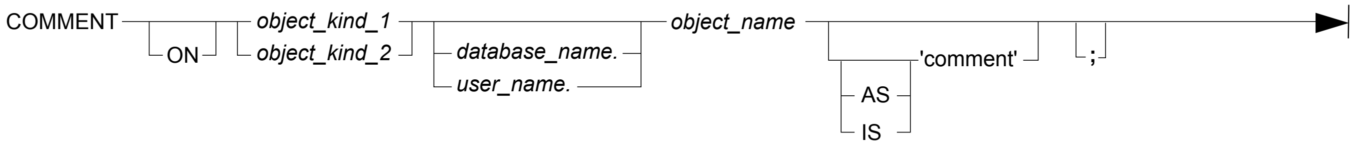
COMMENT (Comment Placing Form)

Purpose

Creates a user-defined description of a user-defined database object or definition in the data dictionary.

This topic describes only the portions of the COMMENT syntax diagram that are specific to Teradata QueryGrid. For information about the other syntax elements in COMMENT (Comment Placing Form), see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

Syntax



Syntax Element

object_kind_2

An optional database object kind specification.

You can specify the following database object kinds to retrieve a comment for the kind of object they represent, but they are optional.

- DATABASE
- FOREIGN SERVER
- TABLE
- USER

If you specify an optional *database_name* with FOREIGN SERVER, the name must be TD_SERVER_DB. You cannot use a *user_name* with FOREIGN SERVER.

All existing rules for COMMENT apply for use with a FOREIGN SERVER object.

The optional comment string is recorded in DBC.TVM.

For more information about using COMMENT (Comment Placing Form), see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

CREATE AUTHORIZATION and REPLACE AUTHORIZATION

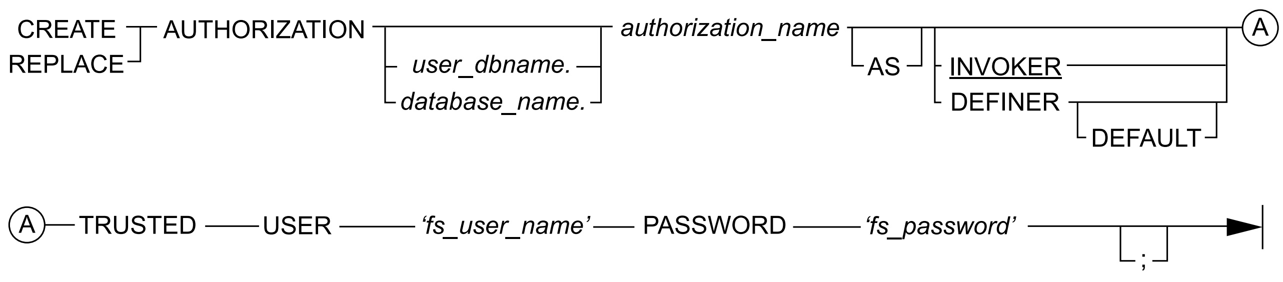
Purpose

Creates or replaces an authorization object in Teradata Database. The authorization stores credentials for a user account that exists on a remote platform. The credentials need only be valid on the platform specified in the foreign server object; they do not need to be valid on the Teradata Database or on its underlying operating system. When you specify TRUSTED in the CREATE or REPLACE AUTHORIZATION statement, Teradata Database does not validate the credentials.

For Teradata QueryGrid, an authorization object is used by a foreign server object to log into a remote platform using credentials that are valid on the remote platform. When a Teradata user makes a request that uses the foreign server, the foreign server object provides the credentials from the authorization object to the target platform for authentication. This allows any part of the request that runs on the remote platform to use the context, privileges, and access control granted to the remote platform user account. For example, if the foreign server connects to a server protected by Kerberos, then the associated authorization object must contain credentials for the user account on that server.

The syntax table describes only the portions of the CREATE AUTHORIZATION and REPLACE AUTHORIZATION syntax diagram that are specific to Teradata QueryGrid. For information about the other syntax that you can use with CREATE AUTHORIZATION and REPLACE AUTHORIZATION, see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

Syntax


CREATE
REPLACE
AUTHORIZATION
user_dbname.
database_name.
authorization_name
AS
INVOKER
DEFINER
DEFAULT
A
A
TRUSTED
USER
'fs_user_name'
PASSWORD
'fs_password'
;

Syntax Elements

database_name.
user_dbname.

Optional name of the location where the authorization is to be stored.

The default location that is used changes based on whether DEFINER or INVOKER is specified. The following rules apply to specifying DEFINER or INVOKER:

- If you specify DEFINER, the database or user you specify must be the containing database or user for the foreign server, UDF, table UDF, method, or external SQL procedure. If no location is specified, the authorization is created in the database that contains the foreign server objects (TD_SERVER_DB).
- If you specify INVOKER, the *database_name* or *user_dbname* you specify must be associated with the session user who will be sending requests to the foreign server. If no location is specified, the authorization is placed in the user database of the creator of the authorization.

authorization_name

Name for the authorization object. This name must be unique within the database in which it is stored.

INVOKER
DEFINER

- If you specify INVOKER TRUSTED, or if you specify TRUSTED alone, Teradata creates the authorization object in the database of the user who creates the object. This syntax makes the authorization available only to those with privilege to the user database.
- If you specify DEFINER TRUSTED or DEFINER DEFAULT TRUSTED, then Teradata creates the authorization object in the database that contains the object that is using the authorization; for a foreign server this is the TD_SERVER_DB database. This syntax makes the authorization globally available.

TRUSTED

A keyword used to specify that the credentials are to be encrypted and stored as database objects.

When using an authorization object, you must use the TRUSTED security type for the Teradata QueryGrid Teradata connector.

You cannot use TRUSTED authorizations in CREATE or REPLACE UDF or XSP statements.

'*fs_user_name*'
'*fs_user_name@realm_name*'

The name of the credential on the remote platform to be used by the foreign server.

The user name for the authorization object can consist of a user name alone or a user name and the name of the Kerberos realm. When only the user name is specified, the default realm specified in the krb5.conf file is used for kinit.

If you include the Kerberos realm name, then the default realm is ignored and the realm that you specify in the authorization object is used for kinit. You should include a Kerberos realm name if you have one of the following situations:

- You want users to be able to connect to multiple Kerberized clusters, each from a different realm, from the same Teradata system.
- You want users to be able to connect to a Kerberized cluster where the authentication realm is different from the service realm.

Note:
If used with the name of the Kerberos realm, the *user_name* portion cannot also contain an @ sign.

'fs_password'

The password for the credential on the remote platform to be used by the foreign server.

All existing rules for CREATE AUTHORIZATION and REPLACE AUTHORIZATION apply.

For more information about using CREATE AUTHORIZATION and REPLACE AUTHORIZATION, see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

Usage Notes

- An authorization is required only if you are using an external security system such as Kerberos for authentication on the foreign server's target platform. .
- You must use either INVOKER TRUSTED or DEFINER TRUSTED when authentication on Hadoop is performed by an external security system such as Kerberos.
- Use INVOKER TRUSTED when you want to create a one-to-one mapping between the Teradata user and the user on the foreign server's target platform. For example, using the same user name for Teradata and Kerberos.
- Use DEFINER TRUSTED when you want to create a many-to-one mapping between Teradata users and a user on the foreign server's target platform. For example, when you want multiple Teradata users who are making requests to the foreign server to use one Kerberos account on the target platform.
- Username and Password connector properties can be set in the **QueryGrid** portlet. They are used for connector diagnostic checks and for end-to-end queries, if they were not provided by the initiator in the authorization object. For information about these properties, see Connector and Link Properties.
- When you create an authorization for another user using INVOKER TRUSTED, *user_dbname* must be specified. Specify the username associated with the session user who will be sending requests to the foreign server. If you fail to specify *user_dbname*, the authorization will be stored in your user database.
- The authorization takes up no space in the database used to store it.
- If your credentials change on the foreign server's target platform, you must remember to replace the credentials in your authorization object. If you fail to update the invalid information, the next time that you try to reference the foreign server object, you get an error message.
- If you drop an authorization object, keep in mind that it may be used by multiple foreign server objects. You should either drop the foreign server objects or alter them so that they specify a valid authorization object. If you fail to update the invalid information, the next time that you try to reference the foreign server object, you get an error message.

Examples of Creating and Replacing the Authorization

If you plan to use the authorization to authenticate to Kerberos on a foreign server then you must use either INVOKER TRUSTED or DEFINER TRUSTED.

The following two examples establish authorization for the user who invokes the object. The credentials are encrypted and stored as a database object in the user database.

```
CREATE AUTHORIZATION sales AS INVOKER TRUSTED
  USER 'johnson'
  PASSWORD 'Secret' ;
```

```
REPLACE AUTHORIZATION sales AS TRUSTED
  USER 'williams'
  PASSWORD 'topsecret' ;
```

If you want to make the authorization available globally, create the authorization on TD_SERVER_DB using the DEFINER TRUSTED type. If you use DEFINER TRUSTED, as in this example, then the credentials for `proxy_1` are stored in the `remote_system1` authorization that is created in the TD_SERVER_DB database.

```
CREATE AUTHORIZATION TD_SERVER_DB.remote_system1
AS DEFINER TRUSTED USER 'proxy_1'
  PASSWORD 'Global' ;
```

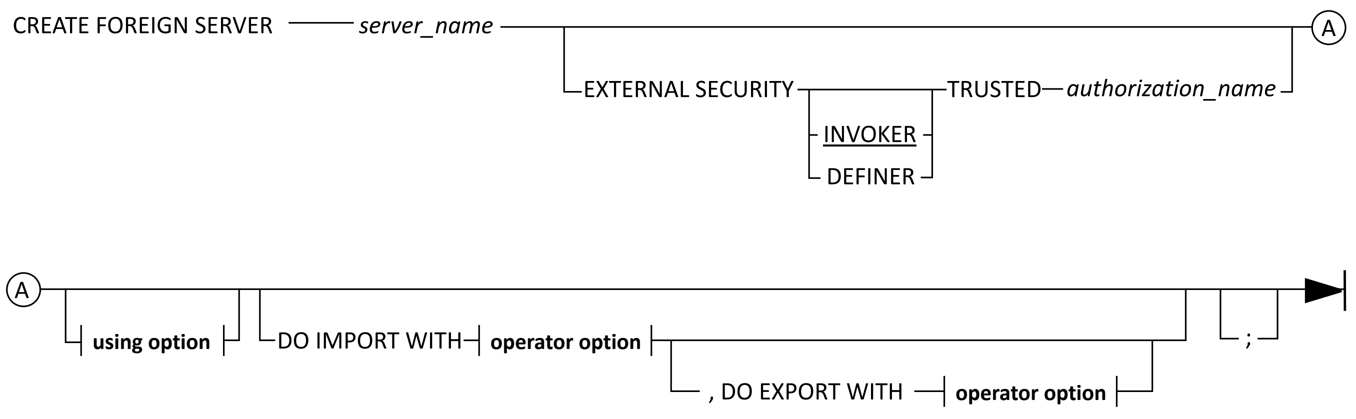
CREATE FOREIGN SERVER

Purpose

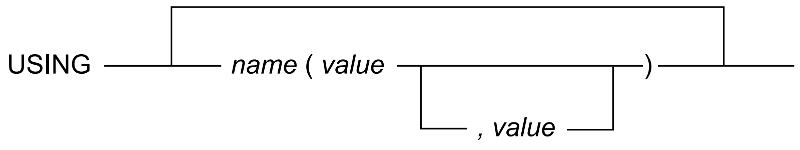
Creates a foreign server object and associates an import table operator and an export table operator with it.

When you create a server object, you can customize it based on its purpose. You can define multiple server objects for the same remote database, each with different characteristics needed by different users.

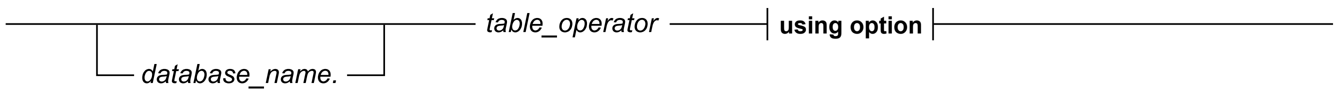
Syntax



using option



operator option



Syntax Elements

server_name

The name given to the foreign server object.

EXTERNAL SECURITY

Associates an authorization object with the foreign server. The authorization stores the encrypted credentials for a user as a database object. The Teradata connector passes the credentials in the authorization to the remote platform identified by the foreign server when the foreign server is accessed.

INVOKER
DEFINER

You can specify either a DEFINER or an INVOKER, but not both. If neither INVOKER nor DEFINER are specified then INVOKER is used by default.

INVOKER is a keyword that indicates that the associated authorization must be present in the user database at the time that the foreign server is accessed.

Note:
The user database is the database that was created for the user in the Teradata system when the user account was created.

DEFINER is a keyword that indicates that the associated authorization must be present in the database that contains the foreign server when the foreign server is accessed.

Note:
The DEFAULT keyword that can be used with DEFINER in CREATE AUTHORIZATION and REPLACE AUTHORIZATION statements is not needed in association with a foreign server.

If you use the DEFINER keyword, the authorization is global for all users who need to query the foreign server.

You must use either INVOKER TRUSTED or DEFINER TRUSTED if the remote platform uses an external security system (such as Kerberos, for example) for authentication.

TRUSTED

A keyword that indicates the associated authorization object was created as TRUSTED.

You must use the TRUSTED security type for the Teradata connector.

authorization_name

Specifies the name of the authorization object to be used when the foreign server is accessed.

USING

USING introduces the name value pair that provides server definition information. You can create a foreign server without a USING clause, but users cannot query a foreign server until you complete the server definition with an import operator and an export operator.

name('value')

The name value pair or pairs define the properties of the foreign server. The name value pairs are defined in the **QueryGrid** portlet. You can create separate named link configurations for unique sets of properties and you can create different versions of the links. For more information, see Connector and Link Properties.

After defining the properties in the **QueryGrid** portlet, use the External Name Value pairs (LINK and VERSION) with the USING clause of CREATE FOREIGN SERVER to associate the name value pair with the foreign server.

The link name value pair is required to create a functioning foreign server object. Version is optional. Additional optional name value pairs may be required to create a foreign server for a specific implementation.

link

A named configuration that references an initiating connector, a target connector, a communication policy, and defines the connector properties to use.

version

The version of the link to use.

You can specify the version of the configuration to use (active or pending) in a Teradata foreign server definition. When pending is specified, then any pending versions of the link, the initiating connector, the target connector, the communication policy, and the fabric are used.

DO IMPORT WITH

Associates an IMPORT function with a foreign server.

DO EXPORT WITH

Associates an EXPORT function with a foreign server.

Required Privileges

You must have CREATE SERVER privilege on the TD_SERVER_DB database to define a foreign server object. If you are associating the server with table operators, you must also have EXECUTE FUNCTION and SELECT privileges on the specified table operators.

Usage Notes

- The target platform of the foreign server object must be running and reachable when you create the foreign server object for it in Teradata Database.
- You can create multiple named foreign server objects that reference the same server using the same link.
- Foreign server object names that are stored in TD_SERVER_DB must be unique.
- Name value pairs in the server area of the syntax apply to the connection to the remote platform and to both of the table operators specified in the IMPORT WITH and EXPORT WITH clauses.
- Server options, names, and name value pairs can appear only once in the CREATE FOREIGN SERVER syntax.

- The order of the DO IMPORT WITH and DO EXPORT WITH clauses in the CREATE SERVER syntax does not matter.
- You must grant SELECT, INSERT, and SHOW privileges on foreign server objects to users who need to query foreign server objects.
- The use of the EXTERNAL SECURITY clause is required when the foreign server's target platform uses LDAP or Kerberos for authentication.

Examples Using CREATE FOREIGN SERVER

The example creates a foreign server using the LINK name value pair. The link and its properties is defined in the **QueryGrid** portlet.

```
CREATE FOREIGN SERVER Presto_1
USING
LINK(‘td1_presto1’)
DO IMPORT WITH TD_SYSFNLIB.QGInitiatorImport,
DO EXPORT WITH TD_SYSFNLIB.QGInitiatorExport;
```

The example creates a foreign server using the LINK and VERSION name value pairs. The link and its properties is defined in the **QueryGrid** portlet. The state of the link (active, pending, or previous) is also set in the **QueryGrid** portlet and referenced in the VERSION clause in the example:

```
CREATE FOREIGN SERVER Presto_1
EXTERNAL SECURITY DEFINER TRUSTED TD_SERVER_DB.Auth_1
USING
LINK(‘td1_presto1’)
VERSION('active')
DO IMPORT WITH TD_SYSFNLIB.QGInitiatorImport,
DO EXPORT WITH TD_SYSFNLIB.QGInitiatorExport;
```

For more information about connector and link properties, see Connector and Link Properties.

For more information about versions, see Replacing Active Fabrics, Connectors, and Links Versions with Pending or Previous Versions.

DROP FOREIGN SERVER

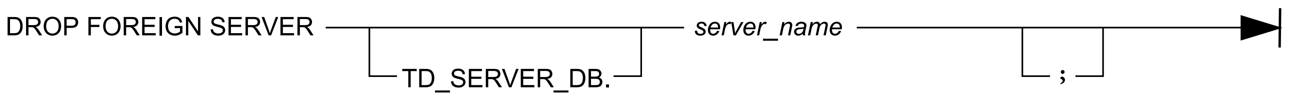
Purpose

Drops a foreign server object from the TD_SERVER_DB database.

In addition to deleting the server object and its associated information from the dictionary tables, all dependent entries on the associated table operators are deleted.

You must have the DROP SERVER privilege on the TD_SERVER_DB database or on the specified foreign server to DROP the foreign server.

Syntax



Syntax Elements

server_name

The name of the foreign server object.

You can also use the following formats for the server name:

- the Unicode Delimited Identifier, such as U&"foreign#005fsv" UESCAPE'#'
- the double-quoted object name, such as "foreign srv1"

TD_SERVER_DB.

The name of the database that stores server objects and their attributes.

Example Using DROP FOREIGN SERVER

The example shows how to drop a foreign server object.

```
DROP FOREIGN SERVER TD_SERVER_DB.Presto1;
```

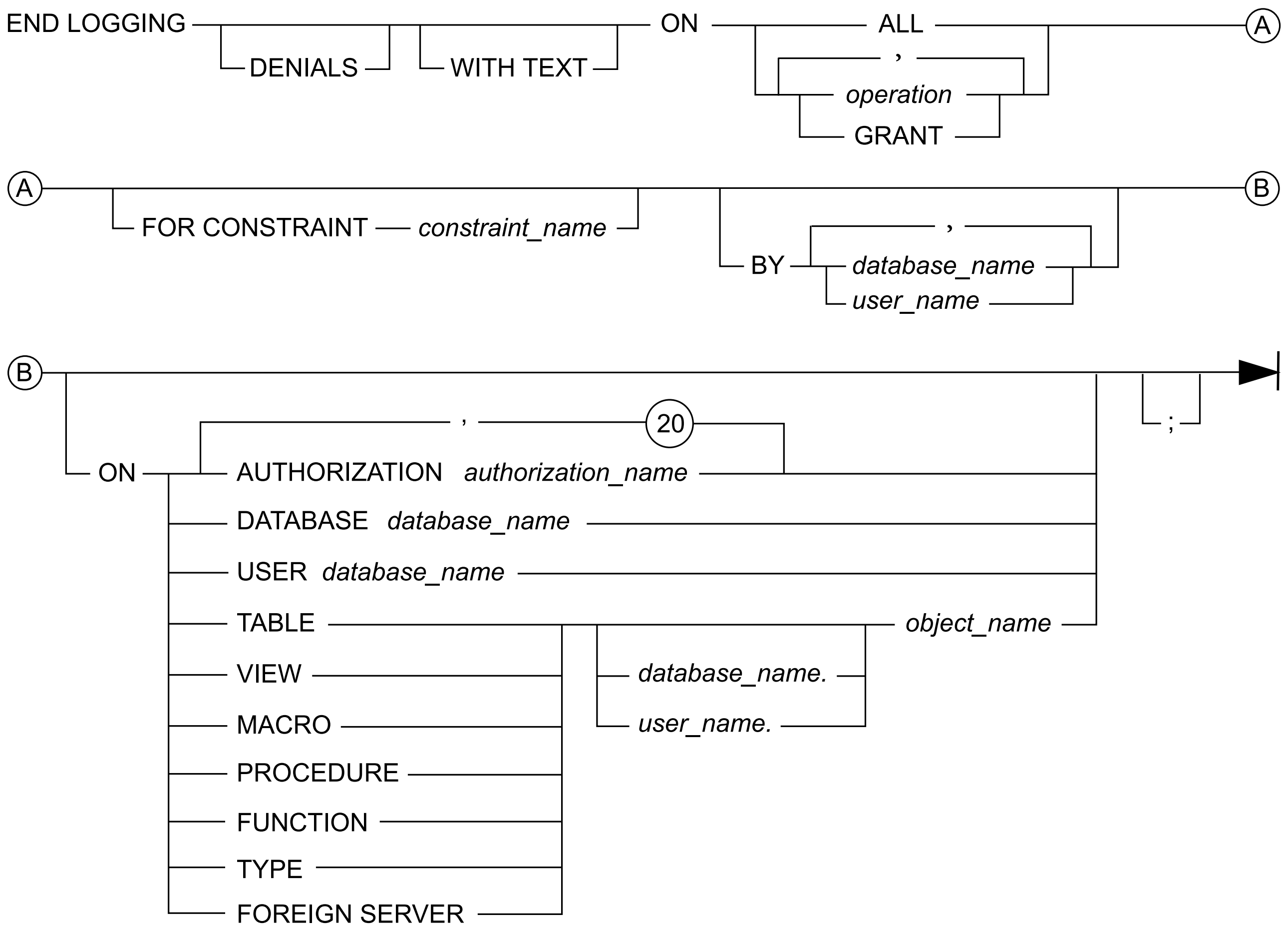
END LOGGING

Purpose

Ends the auditing of SQL requests that started with a BEGIN LOGGING request.

This topic describes only the portions of the END LOGGING syntax diagram that are specific to Teradata QueryGrid. For information about the other syntax that you can use with END LOGGING, see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

Syntax



Syntax Elements

ON *operation*

Indicates the operation for which log entries should no longer be made.

ON FOREIGN SERVER *object_name*

Indicates that the operation for which log entries should no longer be made is access to a foreign server.

You must specify an object name, which is the name of the foreign server. You can optionally specify the name of the containing database, which must be TD_SERVER_DB. You cannot use a *user_name* with FOREIGN SERVER.

For information about using END LOGGING, see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

EXPLAIN

You can use the Teradata QueryGrid connector to query a remote system to request an EXPLAIN plan from the remote system.

For a Presto target system specify the kind of explain you want, either LOGICAL or DISTRIBUTED. To specify the kind, use the **QueryGrid** portlet to configure the target connector with the ExplainKind connector property. See Connector and Link Properties.

Example: Using Teradata to Presto EXPLAIN SELECT

The example generates an explain plan from the remote SELECT query and embeds the result into the query plan on the local Teradata system.

```
EXPLAIN SELECT * FROM p100@tdh111 WHERE c1 > 10;
```

Result:

```
Explanation
---------------------------------------------------------------------------
  1) First, we do an all-AMPs RETRIEVE step executing table operator
     TD_SYSFNLIB.QGINITIATORIMPORT with a condition of ("p100.C1 >= 11").
     < BEGIN EXPLAIN FOR REMOTE QUERY -->
     Remote Queries:
     1. EXPLAIN CREATE TABLE
     qgremote.DEFAULT.TEMP_a1cd2570_d4ef_4b38_b3ed_00000000014e_EXEC AS
     SELECT c1 ,c2 ,c3 ,c4 ,c5 FROM "default"."p100" WHERE C1 >= 11
     [- Output[rows] => [rows:bigint]
     -
     TableCommit[qgremote:QGRemoteTransactionHandle{uuid=280273f9-403c-4cc7
     -b0b5-dc148b36285f}:com.teradata.querygrid.qgc.presto.QGRemoteInsertTa
     bleHandle@e32e808] => [rows:bigint]
     - Exchange[GATHER] => partialrows:bigint, fragment:varbinary
     - TableWriter => [partialrows:bigint, fragment:varbinary]
     c1 := c1
     c2 := c2
     c3 := c3
     c4 := c4
     c5 := c5
     - Exchange[REPARTITION] => c1:bigint, c2:varchar, c3:double,
     c4:boolean, c5:varbinary
     - ScanFilterAndProject[table = hive:hive:default:p100,
     originalConstraint = ("c1" >= 11), filterPredicate = ("c1" >= 11)]
     => [c1:bigint, c2:varchar, c3:double, c4:boolean, c5:varbinary]
     LAYOUT: hive
     c1 := HiveColumnHandle{clientId=hive, name=c1, hiveType=bigint,
     hiveColumnIndex=0, partitionKey=false}
     c2 := HiveColumnHandle{clientId=hive, name=c2, hiveType=string,
     hiveColumnIndex=1, partitionKey=false}
     c3 := HiveColumnHandle{clientId=hive, name=c3, hiveType=double,
     hiveColumnIndex=2, partitionKey=false}
     c4 := HiveColumnHandle{clientId=hive, name=c4, hiveType=boolean,
     hiveColumnIndex=3, partitionKey=false}
     c5 := HiveColumnHandle{clientId=hive, name=c5, hiveType=binary,
     hiveColumnIndex=4, partitionKey=false}
     ]
      <-- END EXPLAIN FOR REMOTE QUERY >
      The size of Spool 1 is estimated with low confidence to be 8 rows
      (33,096 bytes).  The estimated time for this step is 0.15 seconds.
   2) Next, we do an all-AMPs RETRIEVE step from Spool 1 (Last Use) by
      way of an all-rows scan with a condition of ("p100.C1 >= 11") into
      Spool 2 (group_amps), which is built locally on the AMPs.  The
      size of Spool 2 is estimated with low confidence to be 8 rows (
      98,840 bytes).  The estimated time for this step is 0.16 seconds.
   3) Finally, we send out an END TRANSACTION step to all AMPs involved
      in processing the request.
   -> The contents of Spool 2 are sent back to the user as the result of
      statement 1.  The total estimated time is 0.31 seconds.
```

Example: Using Teradata to Presto EXPLAIN INSERT

The example generates an explain plan for the remote INSERT query and embeds the result into the query plan on the local Teradata system.

```
EXPLAIN INSERT INTO p100@tdh111 values(1,'abc',11.1, 1, 1);
```

Result:

```
Explanation
---------------------------------------------------------------------------
  1) First, we do an INSERT into Spool 4.
  2) Next, we do an all-AMPs RETRIEVE step from Spool 4 (Last Use) by
     way of an all-rows scan executing table operator
     TD_SYSFNLIB.QGINITIATOREXPORT with a condition of ("(1=1)") into
     Spool 2 (used to materialize view, derived table, table function
     or table operator p100) (all_amps), which is built locally on the
     AMPs.
     < BEGIN EXPLAIN FOR REMOTE QUERY -->
     Remote Queries:
     1. CREATE TABLE
     qgremote.DEFAULT.TEMP_a1cd2570_d4ef_4b38_b3ed_00000000014f_EXEC AS
     SELECT * FROM "p100" WITH NO DATA
     2. EXPLAIN INSERT INTO "p100" SELECT * FROM
     qgremote.DEFAULT.TEMP_a1cd2570_d4ef_4b38_b3ed_00000000014f_EXEC
     3. DROP TABLE IF EXISTS
     qgremote.DEFAULT.TEMP_a1cd2570_d4ef_4b38_b3ed_00000000014f_EXEC
     [- Output[rows] => [rows:bigint]
     - TableCommit[hive:LegacyTransactionHandle{connectorId=hive,
     uuid=f15a3b4c-89a8-427d-8dcd-c147334ff2ca}:hive:default.p100] =>
     [rows:bigint]
     - Exchange[GATHER] => partialrows:bigint, fragment:varbinary
     - TableWriter => [partialrows:bigint, fragment:varbinary]
     c1 := c1
     c2 := c2
     c3 := c3
     c4 := c4
     c5 := c5
     - Exchange[REPARTITION] => c1:bigint, c2:varchar, c3:double,
     c4:boolean, c5:varbinary
     -
```

```
      TableScan[qgremote:default:temp_a1cd2570_d4ef_4b38_b3ed_00000000014f_e
      xec, originalConstraint = true] => [c1:bigint, c2:varchar,
      c3:double, c4:boolean, c5:varbinary]
      LAYOUT:
      com.teradata.querygrid.qgc.presto.QGRemoteTableLayoutHandle@6029ca45
      c1 := {name = c1, type = bigint}
      c2 := {name = c2, type = varchar}
      c3 := {name = c3, type = double}
      c4 := {name = c4, type = boolean}
      c5 := {name = c5, type = varbinary}
     ]
      <-- END EXPLAIN FOR REMOTE QUERY >
      The size of Spool 2 is estimated with high confidence to be 1 row
      (29 bytes).  The estimated time for this step is 0.03 seconds.
   3) We do an all-AMPs RETRIEVE step from Spool 2 (Last Use) by way of
      an all-rows scan into Spool 3 (Last Use), which is built locally
      on the AMPs.  The size of Spool 3 (Last Use) is estimated with
      high confidence to be 1 row (41 bytes).  The estimated time for
      this step is 0.16 seconds.
   4) Finally, we send out an END TRANSACTION step to all AMPs involved
      in processing the request.
   -> No rows are returned to the user as the result of statement 1.
```

GRANT and REVOKE

GRANT grants one or more explicit privileges on a database, foreign server, user, proxy logon user, table, hash index, join index, view, stored procedure, User-Defined Function (UDF), User-Defined Method (UDM), User-Defined Type (UDT), or macro to a role, group of roles, user, or group of users or databases. REVOKE revokes privileges on the same objects.

There are no changes to existing syntax for Teradata QueryGrid, except that CREATE SERVER and DROP SERVER privileges have been added. These privileges should be granted only to a user, not to a database.

Note:
You are not allowed to GRANT CONNECT THROUGH privilege on the DBC user.

For a syntax diagram and description of the syntax elements that you can use in GRANT and REVOKE, see *SQL Data Control Language*, B035-1149.

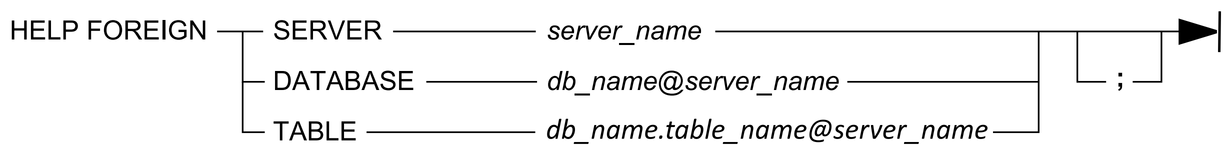
HELP FOREIGN

Purpose

Returns the details of the foreign object that you specify.

- A foreign server object name returns the list of databases accessible on the server.
- The name of a database on a foreign server returns the list of tables in the remote database on the server.
- The name of a table in a remote database on a foreign server returns the list of columns in the remote table on the server.

Syntax



Syntax Elements

SERVER server_name

The name of the foreign server. Displays the databases in the foreign server.

DATABASE db_name@server_name

The name of the remote database, qualified with the name of the foreign server. Displays the tables in the database.

TABLE db_name.table_name@server_name

The name of the remote table, qualified with the name of the foreign server. Displays the column names, data type, and other attributes.

HELP FOREIGN TABLE can be executed with or without the database or schema name (db_name) specified. When the database name is not provided in the query, the currently logged on database or the connector default is used.

Examples That Use HELP FOREIGN

Example: Teradata to Presto HELP FOREIGN SERVER

The example fetches the schema (databases) from the catalog associated with the server.

```
help foreign server QG_Presto1;
```

Result:

```
Databases
--------------------
default
information_schema
finance
marketing
```

Example: Teradata to Presto HELP FOREIGN DATABASE

The example fetches the tables from the specified database.

```
help foreign database "default"@QG_Presto1;
```

Result:

```
Tables
---------------------
buses
ca120453
jing_part_tab1
prama_part_tab1
tbig_1row
web_sales_part1_orc
```

Example: Teradata to Presto HELP FOREIGN TABLE

The example fetches the column information of the specified table.

Note:
HELP FOREIGN TABLE can be executed with or without the database or schema name specified. When the database name is not provided in the query, the currently logged on database or the connector default is used.

```
help foreign table tbig_1row@QG_Presto1;
```

Result:

```
  Column | Type | Comment
————-——-----+—————-----+————-
id | bigint |
firstname | varchar |
lastname | varchar |
gender | varchar |
age | bigint |
```

Required Privileges

You must have ANY privilege granted on the server object to display the output from HELP FOREIGN. The ANY privilege is used for a HELP or SHOW statement for which at least one privilege, but no specific privilege, is required. For more information, see the Privilege Dictionary in *SQL Data Control Language*, B035-1149.

INSERT Syntax and the Teradata Connector

You can export data to a remote system by using an INSERT statement to place data into an existing table. The table can be empty or it can contain data. If the table already contains data, the exported data is appended to the existing table's data.

If no database is specified for the remote database, the current session database or the connector default database is used, depending on the remote connector type and user mapping.

Example: Teradata to Presto INSERT

The example shows an export request initiated on Teradata using the foreign server object (`new_cardata@QG_Presto1`) to insert data from a Teradata table into a table on a remote Presto system.

```
INSERT INTO new_cardata@QG_Presto1 SELECT * FROM td_cardata;
```

SELECT Syntax and the Teradata Connector

You can use the foreign server grammar in the form of *database_name.table_name@server_name* in the SELECT statements of your queries. If you do not specify a database, it defaults to the current database for Teradata-to-Teradata and is used as the default or overridden database provided in the connector or link properties for Teradata-to-Presto. You can use this foreign server grammar in joins and any other place that you reference a normal table, including views, macros, and stored procedures.

Logical expressions such as AND, OR, NOT, >, <, and so on, are supported. Standard Teradata limits, such as returning up to 2048 columns, apply. Row size must be less than 64K.

For a list of the supported data types, see Data Type Mapping for Teradata QueryGrid Connectors.

Query processing is pushed down to the remote system, based on the remote system's capabilities and the local system's optimization. For example, if you query a remote Presto system from a Teradata system, the Teradata optimizer decides the pushdown predicates, which are then evaluated for final remote query pushdown based on what is supported by the remote system.

Example: Teradata to Presto SELECT

The example initiates a request from Teradata to select data from a table on the remote Presto system, using the foreign server object (`presto1`) and the remote Presto table (`cardata`):

```
SELECT CAST(Price AS DECIMAL (8,2))
       , mileage
       , CAST(make AS VARCHAR(20))
       , CAST(model AS VARCHAR(20))
FROM cardata@presto1 WHERE make=’Buick’;
```

Result:

```
       price     mileage  make    model
       --------  -------  ------  --------------------
       17314.10  8221     Buick   Century
       17542.04  9135     Buick   Enclave
```

Example: Teradata to Presto SELECT from Foreign Table

The example initiates a request on Teradata using the foreign table push down query to fetch data from a remote Presto system. This is a pass through query and will not be parsed on the Teradata end, but might need to be regenerated on the remote server, for example, to qualify table name with catalog name:

```
SELECT * FROM FOREIGN TABLE (SELECT make, model FROM default.cardata
                   where make = 'Buick')@QG_presto1 AS dt;
```

Result:

```
make    model
------  --------------------
Buick   Century
Buick   Enclave
```

Note:
A select from foreign table request can only be initiated by Teradata.

Example: Teradata to Presto SELECT from Foreign Table with an EXPORT Clause

The example shows a foreign table query with an additional export clause that combines the export operation and subsequent join and import operation in a single operator. This example first creates a temporary table (temp) in the remote database, exports the data from the local ut2.datatypes2 table to the temporary table, then runs the query on the foreign table to join the temporary table and the remote datatypes1.bigint1 table, and then the result set is brought back Teradata.

```
SELECT * FROM FOREIGN TABLE
  (SELECT datatypes1.bigint1, temp.double1 FROM datatypes1, temp
   WHERE datatypes1.bigint1 > 1)@test EXPORT((select * from ut2.datatypes2) as
temp)
AS ft;
```

Note:
A select from foreign table request can only be initiated by Teradata.

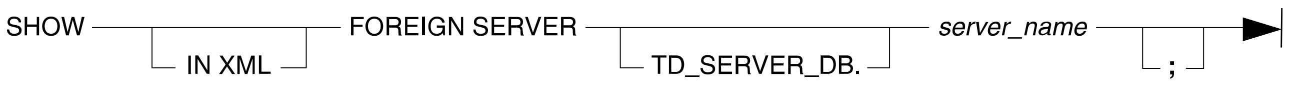
SHOW FOREIGN SERVER

Purpose

Displays the SQL text most recently used to create, drop, or modify the server object.

A SHOW FOREIGN SERVER statement allows you to see a server object definition that contains the name value pairs that the associated table operators use to connect to the foreign server.

Syntax



Syntax Elements

IN XML

To return the report in XML format.

The XML schema for the output produced by this option is maintained in:

http://schemas.teradata.com/dbobject/DBobject.xsd

TD_SERVER_DB.

The name of the database that stores foreign server objects and their parameters.

server_name

The name of the foreign server object.

For the full syntax diagram and information about the other objects that can be used with SHOW, see *SQL Data Definition Language - Syntax and Examples*, B035-1144.

Required Privileges

SHOW FOREIGN SERVER requires SHOW privilege or ANY privilege on the server object to display the output.

Example of Using SHOW FOREIGN SERVER

This example demonstrates the SHOW FOREIGN SERVER that returns XML output.

```
SHOW IN XML FOREIGN SERVER TD_SERVER_DB.presto1;
```

Tuning Query Concurrency on Teradata Database

Viewpoint Workload Designer does not contain a default Teradata Dynamic Workload Manager (TDWM) rule to limit query concurrency for the import and export Teradata QueryGrid table operators (such as QGInitiatorImport, QGInitiatorExport, QGRemoteImport, and QGRemoteExport). It is considered a best practice to create a custom rule in TDWM to set the concurrency limit for these operators. Teradata recommends that you use a range of 6 to 8 concurrent queries under typical circumstances. You can increase this limit to 50 or more queries, but you should keep in mind that when you increase concurrency, there is a cost in terms of query latency. If a rule contains multiple objects of the same type, then the objects are logically OR'd together. It is considered a best practice to not include objects other than functions when you create throttle rules for the connector table operators.

For information about TASM throttle rules, refer to *Teradata Workload User Management Guide*, B035-1197.

For more detailed information about using Viewpoint Workload Designer, see the *Teradata Viewpoint User Guide*, B035-2206.

To create a system throttle that sets the total concurrency limit for QGInitiatorImport, QGInitiatorExport, QGRemoteImport, and QGRemoteExport table operators, follow this procedure:

1. From the **Workload Designer** view, click + to create a new ruleset.
2. Enter the name of the ruleset and optionally, a description, for the ruleset. Click **Save**.
3. Click **Throttles**.
4. Next to **System Throttles**, click the +.
5. On the **General** tab, enter a name and optionally, a description, for the throttle.
6. Click **Save**.
7. Click the **Classification** tab.
8. Select **Target** from the list and click **Add.**
9. Select **Function** from the Target Type list and select **TD_SYSFNLIB** from the database list.
10. Enter the name of the function and click **Include.** The name can be exact or you can use an asterisk (*) as a wildcard to indicate any number of characters. For example, you can use `QGInitiator*` to create a single throttle that matches both `QGInitiatorImport` and `QGInitiatorExport`.
11. Click **OK**.
12. Click **Save.**
13. Click the **State Specific Settings** tab.
14. Under **Default Settings**, click the box and enter the number of queries you want to permit simultaneously.
15. Select either **Delay** to place queries exceeding the limit in a delay queue, or select **Reject.**
16. Click **Save.**

Postrequisite

You'll need to activate the ruleset to make the concurrency limit take effect. From the **Workload Designer** view, click ▾ next to the ruleset that you want and select **Make Active**.

Using APIs to Monitor Queries Between Teradata and a Foreign Server

To monitor the data transfer between Teradata and a foreign server by a user's request, you can use the following APIs:

- PM/API MONITOR SESSION

  Note:
  If you are using the MONITOR SESSION request, set the *mon_ver_id* to 11, where *mon_ver_id* is the monitor software version ID field for the current release.
- Open API MonitorMySessions
- Open API MonitorSession

These APIs return the following field/column values:

Field Name/Column Name	Description
ReqTblOpBytesIn	The total number of bytes transferred into Teradata Database from a foreign server for the current request through one or more table operators. **Note:** The request may involve one or multiple table operator executions. The ReqTblOpBytesIn output parameter shows bytes transferred across all invocations within the request.
ReqTblOpBytesOut	The total number of bytes transferred out of Teradata Database and into a foreign server for the current request through one or more table operators. **Note:** The request may involve one or multiple table operator executions. The ReqTblOpBytesOut output parameter shows bytes transferred across all invocations within the request.

CHAPTER 5

Using the Presto Connector With Teradata QueryGrid

Using the Presto Connector

You can install and configure the Presto connector on independent nodes to access Hadoop data remotely from the HDFS or you can install and configure it on Hadoop nodes.

Ambari is used for the administrative monitoring of Hive.

Presto has its own monitoring system. You can also use the Viewpoint **Query Monitor** portlet to monitor the queries that pass through the Presto connector. For more information, see the *Teradata Viewpoint User Guide*, B035-2206.

For detailed information about configuring the Teradata Distribution of Presto, see the related documentation available from http://www.info.teradata.com: Select **Teradata for Hadoop** > **Teradata Distribution of Presto** > **Teradata Distribution of Presto**.

Presto Connector Limitations

The following describes the limitations on the use of the Presto connector with Teradata QueryGrid.

Presto is limited to queries that can be performed in memory so some queries may not be able to execute in Presto that execute in Hive.

Presto does not support users and access control.

There are some data type conversion limitations from Presto to Teradata. For information about Presto to Teradata data transformations, see Example: Presto to Teradata Data Transformation.

Deploying the Presto qginitiator and qgremote Jar Files

Prerequisite

On the master node of the Hadoop cluster where Presto is deployed, ensure that the presto-admin utility is installed in `/opt/prestoadmin/`. For more information, see http://teradata.github.io/presto/docs/current/installation/presto-admin/quick-start-guide.html.

After adding the Presto connector to a system, complete the following steps to install the `qginitiator` and `qgremote .jar` files.

1. Log on as an Administrator to the master node of the Hadoop cluster where Presto is installed.

2. Change the directory:

   ```
   cd /opt/teradata/tdqg/connector/tdqg-presto-connector/presto-connector-
   software-version/bin
   ```

   For example:

   ```
   tdh234m1:~ # cd /opt/Teradata/tdqg/connector/tdqg-presto-connector/
   1.0-371_SNAPSHOT/bin
   ```

3. Run **`./install.sh`**.
   The script deploys the .jar files to the `/usr/lib/presto/lib/plugin` directory and invokes the prest-admin utility to restart Presto.
4. Verify successful execution of the `install.sh` script as indicated by lack of displayed errors.
5. Verify that the `qginitiator` and `qgremote` `.jar` files installed to applicable directories:

   ```
   tdh234m1:~ # ls /usr/lib/presto/lib/plugin/qginitiator/
   presto-qginitiator-1.0-371_SNAPSHOT.jar
   ```

   ```
   tdh234m1:~ # ls /usr/lib/presto/lib/plugin/qgremote/
   presto-qgremote-1.0-371_SNAPSHOT.jar
   ```

Setting Up the Presto Catalog Properties File

Prerequisite

On the master node of the Hadoop cluster where Presto is deployed, ensure that the presto-admin utility is installed in `/opt/prestoadmin/`. For more information, see http://teradata.github.io/presto/docs/current/installation/presto-admin/quick-start-guide.html.

You define Catalog properties for the Presto connector in a properties file that is manually created and edited. This is different than the Teradata connector properties, which are defined using the **QueryGrid** portlet.

The Presto catalog properties file references a data source and maintains the set of properties associated with that data source connector. You can have one or more catalogs per connector. The Presto connector can have multiple initiator catalogs, one for each link and version pair. The catalog properties file is analogous to the CREATE FOREIGN SERVER statement for the Teradata connector.

Do the following to configure the Presto catalog properties file:

1. Log on as an Administrator to the master node of the Hadoop cluster where Presto is installed.
2. Create and edit an initiator catalog properties file: `/etc/opt/prestoadmin/connectors/<name.properties>`. For example, create and edit a file named `tdh234m1sdld0461_active.properties`. Add the following content to the file:

   ```
   connector.name=qginitiator
   qginitiator.linkName=tdh234m1sdld0461
   qginitiator.version=active
   ```

3. Execute `presto-admin connector add <name>` to deploy the properties file on all nodes. For example:

   ```
   presto-admin connector add tdh234m1sdld0461_active
   ```

4. Restart the Presto cluster. Run:

   ```
   presto-admin server restart
   ```

5. From Presto CLI use the SHOW CATALOGS command to verify the newly created properties file exists. Run:

   ```
   tdh234m1:~ # presto-cli --server localhost:8090
   presto:testuser> show catalogs;
   ```

   Result:

   ```
   Catalog
   -------------------------
    hive
    qginitiator
    qgremote
    system
    tdh234m1sdld0461_active
    tpch
   ```

SQL Syntax for the Presto Connector

Teradata QueryGrid supports the following SQL statements from a Presto connector node:

- SELECT
- INSERT
- SHOW SCHEMAS, SHOW TABLES, SHOW COLUMNS, SHOW CATALOGS
- DESCRIBE

SQL syntax used with the Presto connector differs from Teradata SQL syntax. One notable change is that you must qualify a table with its catalog, for example: `QGNG.salesdb.sales`. `QGNG` is the catalog name, `salesdb` is the schema name, and `sales` is the table name.

The names of catalogs, schema, tables, and columns are not modified by Teradata QueryGrid unless either the local system or the remote system is unable to support the name in its native format. This includes uppercase, lowercase, and mixed case, as well as multi-byte names in language character sets.

SELECT Syntax and the Presto Connector

You can use the foreign server grammar in the form of *catalog_name.schema_name.table_name* in the SELECT statements of your queries. If you do not specify a database, it defaults to the database that is configured for the connector properties in the Viewpoint QueryGrid portlet.

You can use this foreign server grammar in joins and any other place that you reference a normal table, including views, macros, and stored procedures.

Logical expressions such as AND, OR, GT, LT, LE, GE, EQ, ISNULL, IS NOT NULL, IN, and NOT IN are supported pushdown predicates.

For data type mappings from Teradata to Presto and Presto to Teradata, see QueryGrid Data Transformation .

Example: Presto to Teradata SELECT

The example shows an import initiated on Presto to fetch data from a remote Teradata system. The query predicates to be pushed are provided to the connector in the Constraint class by the Presto server and used for pushdown to the remote Teradata system.

```
SELECT MAKE, MODEL FROM QG_TD1.DB1.TD_CARDATA WHERE MAKE = 'BUICK';
```

```
make    model
------  --------------------
Buick   Century
Buick   Enclave
```

Where `QG_TD1` is the catalog used to reference the Presto-to-Teradata link. `DB1` is the database or schema name on the remote system, and `TD_CARDATA` is the table on the remote system.

INSERT Syntax and the Presto Connector

You can use Presto to export data to a remote system by using an INSERT statement to place data into an existing table. The remote table can be empty or it can contain data. If the remote table already contains data, the exported data is appended to the existing table.

Example: Presto to Teradata INSERT

The example shows an export request initiated on Presto to insert data into a remote Teradata Database table.

```
INSERT INTO QG_TD1.DB1.TD_CARDATA SELECT * from hive.default.cardata;
```

`QG_TD1` is the catalog used to reference the Presto-to-Teradata link, `DB1` is the Teradata Database, and `TD_CARDATA` is the table to insert to on Teradata Database.

SHOW Syntax and the Presto Connector

The Presto SHOW statement provides the details of schemas/databases, tables, and table columns on local or remote systems. The SHOW CATALOGS statement lists the contents of the Presto catalog. The following statements are available:

- SHOW SCHEMAS FROM <catalog_name>;
- SHOW TABLES FROM <catalog_name>.<schemaname>;

- SHOW COLUMNS FROM <catalog_name>.<schemaname>.<tablename>;
- SHOW CATALOGS;
- DESCRIBE <tablename>;

The Presto DESCRIBE statement is an alias for the Presto SHOW COLUMNS statement and returns the same information.

The Presto SHOW statement is equivalent to Teradata's HELP FOREIGN statement used when Teradata is the initiator. To see examples of HELP FOREIGN, see Examples That Use HELP FOREIGN.

Example: Presto to Teradata SHOW SCHEMAS

The example fetches the schema (databases) from the remote Teradata system. `QD_TD1` is the catalog used to reference the Presto-to-Teradata link.

```
presto:default> SHOW SCHEMAS FROM QG_TD1;
```

Example: Presto to Teradata SHOW TABLES

The example fetches the list of tables from the specified database (`<schemaname>`) on the remote Teradata system.

```
presto:default> SHOW TABLES FROM QG_TD1.<schemaname>;
```

Example: Presto to Teradata SHOW COLUMNS

The example fetches the column information from the specified table (`<schemaname>.<tablename>`) on the remote Teradata system.

```
presto:default> SHOW COLUMNS FROM QG_TD1.<schemaname>.<tablename>;
```

Note:
The Presto DESCRIBE <tablename> statement can be used instead of SHOW COLUMNS.

Example: Presto SHOW CATALOGS

The example lists the registered catalogs. It does not invoke a Teradata QueryGrid query.

```
presto:default> SHOW CATALOGS;
```

Result:

```
   Catalog
----------------
 hive
 qginitiator
 qgremote
 system
 QG_TD1
 tpch
```

Note:

The Presto-to-Teradata catalog called QG_TD1 is listed as well as the rest of the registered catalogs.

CHAPTER 6
Teradata QueryGrid Stored Procedures, Functions, and Table Operators

Introduction

Teradata supplies several table operators to import and export data to and from remote systems:

- QGRemoteImport
- QGRemoteExport
- QGInitiatorImport
- QGInitiatorExport

Teradata also provides the QGExecuteForeignQuery table operator to enable DDL and DCL actions on a remote system, such as CREATE, ALTER, DROP, GRANT, REVOKE, and so on. QGExecuteForeignQuery is invoked by ExecuteForeignSQL when the user runs a query using ExecuteForeignSQL.

These table operators are activated from DIP.

ExecuteForeignSQL is a stored procedure that provides a simple interface for the users to execute basic SQL queries. ExecuteForeignSQL and an install script are provided as part of the Teradata QueryGrid connector package.

ExecuteForeignSQL

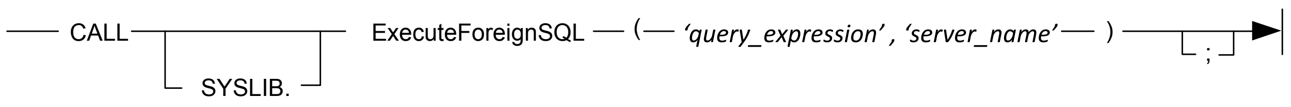
Purpose

The ExecuteForeignSQL stored procedure provides an interface for executing basic SQL queries on the foreign server. For example, you can use ExecuteForeignSQL to create or drop a table in a database on the foreign server.

ExecuteForeignSQL passes all SQL queries through, but it does not return results sets, so if you use it to execute a SELECT or HELP statement, you do not see any results.

ExecuteForeignSQL and an install script are provided as part of the Teradata QueryGrid connector package. The DBA manually executes the script to install ExecuteForeignSQL and then grants the appropriate privileges. Before you use ExecuteForeignSQL you create a foreign server using the QGExecuteForeignQuery table operator in the DO IMPORT WITH clause.

Syntax



Syntax Elements

SYSLIB.

The name of the database where the stored procedure is located.

query_expression

A valid Teradata SQL expression.

The query is passed through unparsed to the foreign server.

server_name

The name of the foreign server.

Usage Notes

ExecuteForeignSQL provides secure execution by using an embedded table operator along with Trusted Sessions to handle logon credential verification.

A DBA can selectively GRANT the privilege to use ExecuteForeignSQL.

To call ExecuteForeignSQL, you must first create a foreign server object that specifies the use of the TD_SYSFNLIB.QGExecuteForeignQuery table operator:

`DO IMPORT WITH TD_SYSFNLIB.QGExecuteForeignQuery;`

You can only associate one import and one export operator at a time with a foreign server, so to run ExecuteForeignSQL on an existing foreign server that is being used for queries, you can create a separate foreign server object with a different name to use for running ExecuteForeignSQL. You must have EXECUTE FUNCTION and SELECT privileges on this operator.

If you reference a table name, you must prepend it with the schema name.

SQL queries that return result sets, such as SELECT or HELP, are executed on the remote system, but the resultant rows are not displayed. The query may continue to use CPU, memory and I/O resources on the remote system.

ExecuteForeignSQL uses a proxyuser, so the execution of commands on behalf of DBC is not supported.

ExecuteForeignSQL can be used with Kerberized clusters.

Example: Using ExecuteForeignSQL

Before you can call ExecuteForeignSQL, you must create the foreign server object and specify `DO IMPORT WITH TD_SYSFNLIB.QGExecuteForeignQuery`, as shown in the example below.

```
CREATE FOREIGN SERVER <fs_name>
EXTERNAL SECURITY DEFINER TRUSTED TD_SERVER_DB.<auth_name>
USING
 LINK('<linkName>')
 version('active')
DO IMPORT WITH TD_SYSFNLIB.QGExecuteForeignQuery;
```

The example creates a table on the foreign server:

```
CALL SYSLIB.ExecuteForeignSQL(
              'create table tab1(c1 int, c2 string)',
              'QG_Presto1');
    );
```

Where `QG_Presto1` is the name of the foreign server.

Teradata QueryGrid Table Operators and Function

QGInitiatorImport

QGInitiatorImport is the initiator import function that is invoked when the query is initiated from Teradata and needs to access metadata and fetch data from a remote host. This initiator import connector function is used to import data from any remote host.

QGInitiatorExport

QGInitiatorExport is the initiator export function that is invoked when the query is initiated from Teradata to access metadata and export data to a remote host. This initiator export connector function is used to export data to any remote host.

Example: Using QGInitiatorImport and QGInitiatorExport

```
CREATE FOREIGN SERVER Presto_1
USING
 LINK(‘QG_presto1’)
DO IMPORT WITH TD_SYSFNLIB.QGInitiatorImport,
DO EXPORT WITH TD_SYSFNLIB.QGInitiatorExport;
```

QGRemoteImport

QGRemoteImport is the remote import function that is invoked when the query is initiated by a remote host to transfer data to the local Teradata system. This remote import function is used for data transfer from any remote data source.

QGRemoteExport

QGRemoteExport is the remote export function that is invoked when the query is initiated by a remote host to access data from the local Teradata system. This remote import function is used to export data to any remote host.

QGExecuteForeignQuery

QGExecuteForeignQuery is the function that is invoked when a stored procedure is invoked to execute a DDL or DML request on the remote system. This allows you to execute CREATE TABLE, DROP TABLE, GRANTs, and so on, from a local Teradata system on the remote host.

Example: Using QGExecuteForeignQuery

```
CREATE FOREIGN SERVER presto_efssp
USING
 LINK(‘QG_presto1’)
DO IMPORT WITH TD_SYSFNLIB.QGExecuteForeignQuery;
```

APPENDIX A

Using the QueryGrid Portlet

Teradata QueryGrid View

The **QueryGrid** portlet allows you to view, configure, and manage the components that make up your Teradata QueryGrid environment. You can use the portlet to perform the following tasks:

- Install and update Teradata QueryGrid fabric, connector, and node software
- Monitor issues specific to Teradata QueryGrid
- View, add, and edit current versions of Teradata QueryGrid fabrics, systems, data centers, nodes, connectors, links, networks, and communication policies
- Manage versions of Teradata QueryGrid configuration entities: fabrics, connectors, links, communication policies, and networks. These configuration entities can be in an active, pending, or previous state. You can add, delete, and activate previous and pending versions of the configuration entities.
- View current QueryGrid managers and add and edit manager service accounts and settings, including query data retention, summary frequency, session timeout, and log data retention
- Map users on one Teradata QueryGrid system to users on another system so that they can submit queries that execute on a remote system

The Teradata QueryGrid portlet contains the following views:

Issues

Displays the details of issues that are specific to Teradata QueryGrid managers and systems; for example, manager offline, node missing, node offline, and nodes running the wrong package version, and so on.

Fabrics

Displays details about the fabrics, connectors, and links that are configured. Allows you to add and activate a pending version of fabrics, connectors, and links, edit and delete versions of fabrics, connectors, and links, view the node status of a fabric or a connector, disable and enable a connector, perform a diagnostic check on a connector or a link, perform a bandwidth check on a link, and view the driver status of a connector.

Connector software is installed on all nodes, but only the driver nodes are used to initiate queries on the target system. A driver runs on one or more target nodes. It is responsible for invoking the remote connector method that initiates the metadata and data execution queries on target.

Systems

Displays details about the systems, nodes, connectors, and links that are configured and allows you to edit, add, and delete systems and add, view, and delete nodes.

Data Centers

Displays details about the data centers, systems, and managers that are configured and allows you to add, edit, and delete data centers.

Networks

Displays details about the networks that are configured, and allows you to edit and delete versions of networks, add and activate a pending version of a network, and view the details of links which use a network.

Communication Policies

Displays details about the communication policies and links that are configured, and allows you to edit and delete versions of communication policies, add and activate a pending version of a communication policy, and view the details of links associated with a communication policy.

User Mappings

Displays details about the user mappings that are configured and the users and links that are involved, and allows you to edit, add, and delete mappings, and edit and delete the users that are in them.

Software

Displays details about the uploaded Teradata QueryGrid software packages and allows you to upload and delete connector, fabric, and node software. After upload the software packages are available to install or upgrade.

Managers

Displays details about the available Teradata QueryGrid managers and allows you to add, edit, and delete service accounts and settings, including query data retention, summary frequency, session timeout, and log data retention.

Accessing QueryGrid Portlet

Prerequisite

The **QueryGrid** portlet is accessed from Teradata Viewpoint.

The user must be granted permission to access the **QueryGrid** portlet. The Teradata Viewpoint Administrator uses the **Roles Manager** portlet to assign portlet permissions to the roles. Setting the Edit Configurations permission enables the role to edit the Teradata QueryGrid configuration, run a diagnostic check, run a bandwidth test, and delete active issues.

1. Add the **QueryGrid** portlet to your Viewpoint page. For instructions on how to add it, see Adding Portlets in *Teradata Viewpoint User Guide*.
2. Select **QueryGrid** from your Viewpoint page.

Summary Table Controls

In the **QueryGrid** portlet, data is displayed in a summary table. You can set which table columns to display and export the data to a .csv file for further analysis and formatting. If available, you can also filter the displayed content.

Configuring Columns to Display

Use the **Configure Columns** dialog box to reorder columns. You can also resize the columns in the table.

1. Click ▾ in the table header and select **Configure Columns**.
2. In the **Configure Columns** dialog box, select the check boxes of columns to display.

   Mouse over the column name to see a description of the data the column displays.
3. [Optional] To lock the column position, click 🔒 next to the column name.

   The columns at the top of the list can be locked in the table to remain on the left when scrolling horizontally.
4. [Optional] Click **Set Threshold** , type a threshold value, and click **OK**.

   Threshold values can be set only in certain portlets, and only for certain columns.

   Qualifying data is highlighted in the table.
5. [Optional] Click ≡ and drag the row to reorder the column.
6. Click **OK**.
7. [Optional] In the table, drag the column heading border ←‖→ in either direction to resize the column.

Filters and Sorting

Filters allow you to display only rows that match your filter criteria. You can narrow the search further by filtering on multiple columns. You can also:

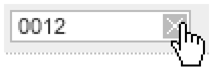
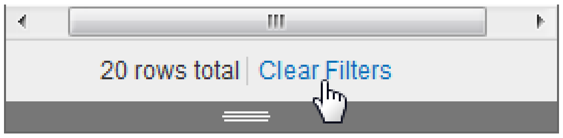
- Use greater than or less than symbols when filtering columns with numeric values. Exact matches may not produce the expected results due to rounding of numbers containing decimals.
- Use wildcard characters or symbols in the filter to include number or word variations in the filter match.
- Use the filter bar to display only specific states, severity levels, or time periods.

Sorting allows you to change the order of rows in a table based upon criteria in a column and applies across all pages of the table. Sort on a column by clicking the column header. A second click reverses the sort order. Sort on two columns consecutively by sorting the first column and then using **Ctrl+Click** once in the column header of the second column. Using **Ctrl+Click** one time sets the secondary sort order. Primary sort order is indicated by a single arrow, and secondary sort order is indicated by a double arrow. The secondary sort order is removed when the primary or secondary sort column has been modified. After modification, you see a single arrow to indicate the secondary sort order has been removed.

Clearing Filters

You can clear the column filters from the table.

1. Do one of the following:

Option	Description
Clear individual column filters	Click ☒ on the filter box.  0012
Clear all column filters	Click ▾ in the table header and select **Clear Filters**.
Clear all column filters	Select **Clear Filters** at the bottom of the portlet.  20 rows total | Clear Filters

Exporting Table Data

You can export data to a .csv file for further analysis and formatting.

The format for the time, date, and some numeric values differs in the view and exported .csv file.

1. Click ▾ in the table header and select **Export**.
2. Save the file using the browser options.
   The file is saved to your download area or to a location that you specify, depending on the browser settings.

Issues View

The **Issues** view allows you to review possible problems with Teradata QueryGrid. Some of these issues might require that you take action. The following types of issues are reported:

- Teradata QueryGrid component has gone offline
- Teradata QueryGrid manager has failed
- Teradata QueryGrid manager backup failed
- Teradata QueryGrid manager is running out of disk space
- One or more Teradata QueryGrid managers are running different versions of the Teradata QueryGrid manager package

- One or more nodes on the system are missing the connector package
- System time is incorrect on a node or a Teradata QueryGrid manager

Each condition has an associated severity, either critical or warning.

You can view the history of an issue from the **View History** link at the top right of the **Issues** table.

For information about viewing the history of an issue, see Viewing Issues History.

Table Actions

Clear Filters removes any content in the filter boxes.

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Row Actions

Delete allows you to delete an issue.

Viewing Issues History

1. From the **Issues** view, click **View History**.
2. If you do not want to view the issues history for the last week (default), click next to the time range and select another time range option from the list.

   You can choose to view issues from the following time periods:

   - Last 1 hour
   - Last 1 day
   - Last 1 week (Default)
   - Last 1 month
   - Last 3 months

Issues Details View

Issues details consists of a read-only text box that displays the entire error message that was returned for this issue. Select the issue to view the entire message.

Fabrics View

Fabrics represent a set of interconnected nodes, all of which run compatible versions of Teradata QueryGrid software over the same port. A fabric organizes the Teradata QueryGrid components into a single environment. Fabrics enable different connectors to communicate with one another.

Connectors and links are associated with a Fabric. Connectors are software that connect a data source, such as Teradata or Presto, to the fabric. Links define what connectors can access each other and how data is transferred between the connectors.

The connectors associated with a fabric are of different types, for example, Presto or Teradata, and each runs a version of the Teradata QueryGrid connector software. There is an initiating connector, which is the point from which a query originates, and a target connector on the target system. When you add connectors, you can also define optional connector properties, which are specific to each type of connector.

Links are named configurations that include an initiating connector, a target connector, and a communications policy. When you add links, you can also define optional link properties, specific to the initiating or target connector and specific to each type of connector, such as Presto or Teradata. Note, link properties override connector properties. Links can also optionally contain initiator or target networks. You can reference particular links in Teradata QueryGrid foreign server definitions to simplify configuring foreign servers.

You can view fabric details and sort on the column headers to find the details you want.

Table Actions

Clear Filters removes any content in the filter boxes.

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

The row actions that are available on each row vary depending on whether the fabric is a previous, pending, or active version.

Row Actions

Activate allows you to change a pending or previous version of a fabric to the current (active) version. The previous fabric, if any, is deleted, and the version of this fabric that was current becomes the previous version of this fabric.

Add Pending Version allows you to create a pending version of the selected fabric that you can edit and test.

Delete allows you to delete a fabric.

Edit allows you to change the name and description of a current fabric version. If you need to change the port or software version that is used in a fabric, create a pending version of a fabric, wait for all the nodes to come online, test it, and then activate it.

View allows you to view the details of a previous fabric version.

View Node Status allows you to view the status of the nodes that are associated with a fabric.

Adding a Fabric

Add one or more fabrics to organize the components in the Teradata QueryGrid environment so that you can manage them more easily.

1. From the **Fabrics** view, click + next to **Fabrics**.
2. Type a name in the **Fabric name** box, up to 256 characters.
3. Type a description in the **Description** box, up to 1000 characters.
4. Type the port number in the **Port** box.

   Valid values range from 1024 to 65535. Select a value that is not currently in use by the systems you plan to add to the fabric.
5. Do one of the following:

Option	Description
Select an existing version	**a.** From the **Fabric software** list, select a version. **b.** Click **Save**.
Upload a version	**a.** Next to the **Fabric software** list, click +. **b.** Browse to the file and click **Upload**. **c.** From the **Fabric software** list, select the uploaded version. **d.** Click **Save**.

After you create a fabric, you need to make it usable by adding connectors and links to it.

Editing a Fabric

Edit a fabric when you need to change the name or description of the fabric.

1. From the **Fabrics** view, click ▼ next to the fabric row that you want to edit and select **Edit**.

   Only **Fabric name** and **Description** can be edited.

Viewing Fabric Details

1. From the **Fabrics** view, select the fabric that you want to see details about.
2. Click ▼ next to the fabric that you want and select **View Node Status** to see the details of the nodes that are used in this fabric.

Connector and Link Properties

The following types of Name Value Pairs (NVP) can be configured for connectors and links:

- Configuration Name Value Pairs
- External Name Value Pairs

Configuration Name Value Pairs

Configuration NVP are user configurable settings. The majority of these NVP specify the behavior of the target connector component, configure how data is transformed, or configure the underlying link data transportation layer. A small subset of the NVP affect how the initiator connector performs.

Configuration NVP are configured in Viewpoint using the **QueryGrid** portlet. These NVP can be set for both links and connectors.

Links are named configurations that include an initiating connector and a target connector. In some cases, if you set a link NVP, that was also set in the connector, it overrides the connector setting.

Note:

Some properties are available only for initiating connectors or target connectors.

Name	Default	Description	Connector Type	Initiating or Target connector
Authentication Mechanism	Trusted is the default for Teradata and None is the default for Presto.	Indicates the authentication mechanism used on the target data source. The valid values for the Teradata connector are: Trusted and LDAP. The valid values for the Presto connector are: None and Kerberos.	All	Target
Byte Count Report Frequency	64KB	The frequency with which byte count is updated in DBQL. You can adjust this to a larger value if the update frequency is too resource intensive.	Teradata	Both
Catalog Name	hive	Name of the catalog for the Presto connector.	Presto	Target
Schema Name	default	The name of the schema name for the Presto connector	Presto	Target
Database Name	DBC	Name of the default schema or database for the connector. Note, the default database is not necessarily the current database.	Teradata	Target
Default Binary Size	4096	The default truncation size for the unlimited length VARCHAR types. This is for a Teradata-to-Presto link and is used by the target Presto connector.	Presto	Target
Default String Size	4096	Size at which data imported from or exported to string columns is truncated. The value represents the maximum number of Unicode characters to import, and defaults to 4096 characters. If not set, Teradata QueryGrid silently truncates the string columns at the default value set in default_string_size. This is for a Teradata-to-Presto link and is used by the target Presto connector.	Presto	Target
Disable Pushdown	False	When set to true, disables the pushdown of all query conditions to the target system. Valid values are true and false.	All	Target

Table continued on next page.

Name	Default	Description	Connector Type	Initiating or Target connector
		Certain system-level, session-level, and column-level query attributes, such as casespecific, can affect character string comparison results. These attributes can cause some queries to return incorrect results due to incorrect row filtering on the target system. To avoid incorrect results caused by condition pushdown in situations where the settings on the initiating system do not match the settings on the target system, you can disable the pushdown of all conditions to the target system.		
Enable Async Abort	False	Indicates if PDE asynchronous abort should be enabled. Valid values are true and false.	Teradata	Both
Enable Logging	None	Sets the logging level for the connector or link properties. User level log settings can be explicitly set through the add or edit link page in the **QueryGrid** portlet (select "Enable debug logging for initiating user"). This setting applies to both the initiating and target connector; however, the initiating connector's logging level for the link takes precedence if they are set differently.	All	Both
ExplainKind	None	Defines the flavor of the explain for the EXPLAIN SQL statement for a remote Presto query. Valid values are logical or distributed.	Presto	Target
Keytab	None	Filename with absolute path to the Kerberos keytab file.	Presto	Target
Link Buffer Count	4	Maximum number of write buffers available on a single channel at one time.	All	Both

Table continued on next page.

Name	Default	Description	Connector Type	Initiating or Target connector
		Note: Link Buffer Count overrides the default internal fabric property shmDefaultNumMemoryBuffers.		
Link Buffer Size	73728	The size of the writer buffers to allocate for row handling and message exchange.	All	Both
Link Handshake Timeout	30000	The handshake timeout in milliseconds for the link channel setup.	All	Both
Link Heartbeat Interval	60000	The maximum interval in milliseconds for the heartbeat signal on a channel between the connector and the fabric instance, used for health check status. **Note:** This interval should be greater than Link Handshake Timeout.	All	Both
Password	None	User password.	All	Target
Port	None	The default value for a Teradata connector is 1025. A Presto connector requires a port number and it is typically set to 8080. The default value can be overridden.	All	Target
Presto Reader Task Concurrency	8	The number of parallel concurrent readers in Presto per worker per query.	Presto	Target
Presto Writer Count	8	The number of parallel writers in Presto or concurrent writer threads per worker per query.	Presto	Target
Read Timeout	300000	The number of seconds to wait to read between data packets during data transfer.	All	Both
Realm	None	Kerberos realm	Presto	Target
Replace Unsupported Unicode Character	False	Replaces unsupported Unicode characters with U+FFFD, when importing data by the Teradata connector.	Teradata	Target

Table continued on next page.

Name	Default	Description	Connector Type	Initiating or Target connector
		Note: If you are using Unicode Pass Through characters, do not set this property to true. See the Validate Unsupported Unicode Characters property.		
Response Timeout	1800000	Number of seconds to wait for the final data exec response when all the data has been transferred.	All	Both
Server	None	This property is used to connect to the target database as part of the JDBC connection string.	All	Target
SSL Certificate	None	SSL certificate path for Kerberos.	Presto	Target
Transformformatting	False	[Teradata-to-Presto] Indicates that array list data is formatted appropriately, so that it can be cast directly into a Teradata array column type based on the appropriate data type. A RETURNS clause can cast at the column level, but if this property is set, this link level connector property overrides any value set in the RETURNS clause. When set to true, all array columns are returned in Teradata array string format. When set to false, individual columns can be requested to be returned in Teradata string format in the RETURNS clause.	Presto	Target
Username	Default username for the connector. For the Teradata connector the default is DBC.	Name of the user. This NVP is saved in the Teradata QueryGrid manager configuration and is required when the initiator does not support a mechanism to provide user credentials. The username is also used for connectivity diagnostic checks.	All	Target
Validate Unsupported Unicode Characters	False	Validate unsupported Unicode character when importing data by the Teradata connector.	Teradata	Target

Table continued on next page.

Name	Default	Description	Connector Type	Initiating or Target connector
		Note: If you are using Unicode Pass Through characters, do not set this property to true. See Replace Unsupported Unicode Character.		
Write Timeout	300000	The number of seconds to wait to write between data packets during data transfer.	All	Both

External Name Value Pairs

External NVP are defined for links and for connectors.

- For Teradata, the External NVP are set in the Foreign Server definition. The External NVP must be set up so that the Teradata QueryGrid connector can identify itself with the Teradata QueryGrid manager to obtain the configuration NVP. Additional External NVP may be set up on Teradata to access specific features like name and role mapping, but should be used with care to not conflict with values supplied by Teradata QueryGrid manager.

  The External NVP are called LINK and VERSION. These are configured using SQL commands: ALTER FOREIGN SERVER or CREATE FOREIGN SERVER.
- For Presto, the External NVP are stored in the link properties file. This file is manually edited and then the Presto server is restarted for the new configuration to take effect.

Connectors Tab

Use the **Connectors** tab on the **Fabrics** view of the **QueryGrid** portlet to view the details of existing connectors and to add a connector to the fabric.

The row actions that are available on each row vary depending on whether the connector is a previous, pending, or active version.

Row Actions

Activate allows you to change a pending or previous version of a connector to the current (active) version. The previous connector, if any, is deleted, and the version of this connector that was current becomes the previous version of this connector.

Add Pending Version allows you to adds a pending version of the selected connector.

Delete allows you to delete the selected connector.

Diagnostic Check allows you to run a diagnostic check on the selected connector.

Disable/Enable allows you to disable or enable a connector.

Edit allows you to modify the current or pending version of the selected connector.

View allows you to view the details of a previous connector version.

View Driver Status allows you to view the status of the drivers on the selected connector.

View Node Status allows you to view the status of the nodes on the selected connector.

Adding a Connector

Prerequisite

If you are adding a Presto connector, the `presto-admin` utility must exist on the cluster because the `install.sh` script used to complete the process invokes the utility. The `presto-admin` utility can be located in any directory, but typically it is in `/opt/prestoadmin/presto-admin`.

Queries require a connector to run, so connectors must be added to a fabric. Connectors translate the request language of queries and transform data so that it can be exchanged between different types of systems. Connectors control the execution of queries on target systems and return the results to initiating systems.

1. From the **Connectors** tab on the **Fabrics** view, click + next to **Connectors**.
2. Type a name, up to 256 characters.
3. Type a description, up to 1000 characters.
4. From the **System** list, select a system that you want to add this connector to.
5. Click + to add the allowed OS user that this software will run as.

   This user is typically the same user under which the database runs; for example, for a Teradata Database connector it is the *teradata* user and for the Presto connector it is the *presto* user.
6. Do one of the following:
   - Select a version of connector software from the list.
   - Click + next to **Connector software** to upload a version of connector software that is not in the list, so that the newly uploaded software version appears as the selected item in the list.
7. Depending on the connector software that you chose, click the available properties that you want the connector to use.

   The required properties are listed below **Properties**. If you edit Properties and select an additional property, it is subsequently required, until you unselect it. For more information, see Connector Properties.
8. Under **Driver Nodes**, do one of the following:

Option	Description
Make available on all nodes	a. Click **Make available on all nodes**.
Make available on selected nodes only	a. Click **Make available on selected nodes only**. b. Click **Select Nodes**. c. In the **Available nodes filter** box, type some text and one or more wildcards to get a subset of the available nodes to display in the **Available nodes** list. d. Use the > to move the nodes that you want from the **Available nodes** list to the **Selected nodes** list. e. Click **Save**.

Connector software is installed on all nodes, but only the driver nodes are used to invoke queries on the target system. A driver runs on one or more nodes associated with the system associated with a target connector. A driver loads the connector, reads messages, invokes methods to process requests, and sends responses.

Only a subset of the nodes on a large system need to be driver nodes. Use multiple driver nodes for redundancy and to share the workload required for query initiation. The driver runs on one or more target nodes.

9. [Optional] From the **Associated Viewpoint system** list, select a Viewpoint system to associate with this connector.

   The associated Viewpoint system is used to link Viewpoint monitored systems to Teradata QueryGrid queries in portlets such as Query Monitor.
10. Click **Save**.

Postrequisite

If you are adding a Presto connector, there are manual steps to complete the process. See *Teradata QueryGrid Installation, Configuration, and Upgrade Guide*, B035-5990.

Editing a Connector

Edit an existing connector when you need to change one or more of the following attributes:

- Name
- Description
- OS user name under which it runs
- Version of the connector software
- Connector properties
- Nodes on which the connector is available to initiate queries. Note, the connector software is installed on all nodes of the system, but only driver nodes can initiate queries.
- Viewpoint system used to manage the connector

1. From the **Connectors** tab on the **Fabrics** view, select a connector.
2. Click ▾ next to the connector that you want to edit and select **Edit**.
3. Modify the attributes of the connector that you want to change.
4. Click **Save**.

Viewing Node Status

View node status to see its host name and the system that it is part of. You can also see its status, either online or offline.

1. Do one of the following:
   - From the **Fabrics** view, click ▼ next to the fabric that you want and select **View Node Status**.
   - From the **Connectors** tab on the **Fabrics** view, click ▼ next to the connector that you want and select **View Node Status**.

Viewing Driver Status

View driver status to see its host name and the system that it is part of. You can also see its status, either online or offline.

1. From the **Connectors** tab on the **Fabrics** view, click ▼ next to the connector that you want and select **View Driver Status**.

Running a Connector Diagnostic Check

Run a diagnostic check on a connector to determine whether the target connector is available and whether connector being diagnosed can access the target system using the specified connector properties.

1. From the **Connectors** tab on the **Fabrics** view, click ▼ next to the connector that you want to run a diagnostic check on and select **Diagnostic Check**.
2. Click **Active** or **Pending** to test either the active or the pending version of the connector and the associated fabric.
3. If requested, enter the values of the connector properties.

   If a connector has a property that is required for the diagnostic check and has not yet been entered, it will appear here.
4. Click **Run**.
   The test results display in the **Connector Diagnostic Check** window.

   For more information see Connector Diagnostic Check Metrics.

Links Tab

Use the **Links** tab in the **QueryGrid** portlet to view the details of existing links and to add a link to Teradata QueryGrid.

In addition to adding a link, you can take the following row actions:

▾ **Row Actions**

Activate allows you to make a pending or previous version of the link the active version. Appears only if the version is pending.

Add Pending Version allows you to adds a pending version of the selected link.

Bandwidth Test allows you to run a bandwidth check on the selected link.

Delete allows you to delete the selected link.

Diagnostic Check allows you to run a diagnostic check on the selected link.

Edit allows you to modify the current or pending version of the selected link.

View allows you to view the properties of the link. Appears only if the version is previous.

Adding a Link

Add links to simplify Teradata QueryGrid configuration. Each link specifies an initiating connector and a target connector to use for querying and defines the communication policy and optional connector properties to use.

1. From the **Links** tab on the **Fabrics** view, click + next to **Links**.
2. Type a name, up to 256 characters.
3. Type a description, up to 1000 characters.
4. Select an initiating connector to send the queries.
5. [Optional] Add **Initiating Properties**. Select the desired properties and click **Save**.

   For more information, see Connector and Link Properties.
6. Select a target connector to receive the queries from.
7. [Optional] Add **Target Properties**. Select the desired properties and click **Save**.

   For more information, see Connector and Link Properties.
8. Do one of the following to add an **Initiating network**:

Option	Description
Select an initiating network	**a.** Select a network from the list.
Add an initiating network and select it	**a.** Click + next to **Initiating network** to add the network that you want this link to send queries over. **b.** Type a name, up to 256 characters. **c.** Type a description, up to 1000 characters. **d.** Select the **Matching rules** format, either **CIDR notation** or **Interface name**. You can select interface name and type in the interface name or you can select CIDR notation to use the Classless Inter-Domain Routing (CIDR) designation for the physical network interface instead of an interface name. You can use up to 500 characters. **e.** [Optional] Click + to add more rules. You can click − to remove an existing rule. **f.** Click **Save**.

Option	Description
	g. Select the network you just added for **Initiating network**.

9. Add a **Target network**. Do similar steps performed to add an initiating network.

10. Do one of the following to add a **Communication policy**:

Option	Description
Select a communication policy	**a.** Select a policy from the list.
Add a communication policy and select it	**a.** Click ✚ next to **Communication Policy**. **b.** Type a policy name, up to 256 characters. **c.** Type a description, up to 1000 characters. **d.** Type an integer from 1 and 5 to set the number of data transfer streams that can run simultaneously. **e.** Select the combination of authentication, integrity and encryption checks you want to use. Fill in values, if requested. The checks appear in order of increasing security level, with the choice that provides the greatest level of security listed last. **f.** Click **Save**. **g.** Select the policy you just added for **Communication policy**.

11. [Optional] Select a **User mapping** to use over this link.

User mapping enables a specific user on an initiating system to submit queries as a specific user on a target system.

12. [Optional] Select **Enable debug logging for initiating user** if you have a query or set of queries that are failing or not performing correctly for a particular database user for an unknown reason.

If you enable this option and have the user run the query again, detailed logging is captured by the Teradata QueryGrid manager. Use Info for debugging performance issues and Debug for debugging incorrect or failing behavior. A Teradata Technical Support Specialist can then use this information to troubleshoot the issue.

13. Click **Save**.

Editing a Link

You can edit any of the link properties. You cannot edit a previous version of a link.

1. From the **Links** tab on the **Fabrics** view, click ▼ next to a link and click **Edit**.

   When selecting initiating and target connectors only active connections are available from each list.

Running a Link Diagnostic Check

Run a link diagnostic check to test the connectivity between all the nodes of initiating and target systems, based on the communication policy that is associated with the link. The check reports either pass or fail. If it fails, it reports why it failed.

1. From the **Links** tab on the **Fabrics** view, click ▼ next to the link that you want to run a diagnostic test on and select **Diagnostic Check**.

2. Click **Active** or **Pending** to test either the active or the pending version of the link.
3. Click **Run**.
   The test results display in the **Link Diagnostic Check** window.

Running a Link Bandwidth Test

Run a link bandwidth test to test the bandwidth between all the nodes of initiating and target systems, based on the communication policy that is associated with the link. This test reports an average byte rate between systems and helps to identify bottlenecks.

1. From the **Links** tab on the **Fabrics** view, click next to the link that you want to run a bandwidth test on and select **Bandwidth Test**.
2. Click **Active** or **Pending** to test either the active or the pending version of the link.
3. Click **Initiating to target** or **Target to initiating** depending on the direction of the link data flow that you want to test.
4. Click **Run**.
   The test results display in the **Link Bandwidth Test** window.

   For more information, see Link Bandwith Test Metrics.

Working with Active, Pending, or Previous Versions

Making changes to the Teradata QueryGrid configuration can potentially impact production workloads, so Teradata QueryGrid supports the ability to test configuration changes before those changes are activated. After the configuration change is tested and verified, it can be activated so that it becomes available to production workloads.

Changes to configuration entities (fabrics, connectors, links, communication policies, and networks) are versioned. When a query starts, it resolves which versions of the configuration entities it will use. These versions are passed around to all components executing on behalf of that query, so that all components use the same version of the configuration.

When a configuration change is made, it may take some time for that change to reach all the components of a system. If a query references a version of a configuration entity that has not been propagated throughout the system, the query will fail. Configuration changes should be persisted and tested before being applied to production workloads, to ensure that the configuration change is distributed to all the components of Teradata QueryGrid.

Configurations can be in one of three states: active, pending, and previous. Active is the tested and verified actively running version of the configuration. Pending is used for testing new or modified configurations. Active configurations are put in a previous state when another version of the configuration is activated.

If you replace an active version, the formerly active version becomes the previous version and the former previous version is deleted. If you activate a pending version, the formerly active version that you replaced becomes the previous version. If you activate a previous version, the active version becomes the previous version, and any pending version remains the pending version.

Replacing Active Fabrics, Connectors, and Links Versions with Pending or Previous Versions

1. Click **Fabrics**.
2. Do one of the following:

Option	Description
Fabrics	**a.** Click ▾ next to the pending or previous version that you want to make the active version and select **Activate**.
Connectors	**a.** Click the **Connectors** tab. **b.** Click ▾ next to the pending or previous version that you want to make the active version and select **Activate**.
Links	**a.** Click the **Links** tab. **b.** Click ▾ next to the pending or previous version that you want to make the active version and select **Activate**.

Systems View

A system is the platform (for example, Hadoop, Teradata Appliance, Teradata Aster Appliance) that one or more data sources share, and consists of one or more nodes. A single system can host multiple connectors.

The nodes in the system run the tdqg-node and tdqg-fabric software and are associated with one system only. The fabric software allows the Teradata QueryGrid connectors to communicate with each other in parallel, and to execute, process, and transport data in an efficient manner. There is no restriction on the size of the data that is transported between nodes.

The tdqg-node package manages the connectors and fabrics on the node. Teradata QueryGrid tdqg-node software must be installed on every node in a system.

You can view system details and sort on the column headers to find the details you want.

▾ **Table Actions**

Clear Filters removes any content in the filter boxes.

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

▾ **Row Actions**

Edit allows you to change the system configuration.

Delete allows you to delete a system.

Adding a System

1. From the **Systems** view, click + next to **Systems**.
2. Type a name for the system, up to 256 characters.
3. Type a description, up to 1000 characters.
4. Do one of the following from the **Data center** list:
   - Select the data center where this system is located.

- If the data center where this system is located is not in the list, click ✚ next to the **Data center** list to add the data center.

5. Do one of the following from the **Node software** list:
   - Select the version of the node software that will run on this system.
   - If the version of the node software that runs on this system is not in the list, click ✚ next to the **Node software** list to upload the software version file.
6. Set the **Memory Requirements**:

Option	Description
Query concurrency	Maximum number of Teradata QueryGrid queries supported from the system at the same time. Teradata recommends that customers determine this value based on their daily workload and memory requirements for Teradata QueryGrid. 50 is the default recommendation. Valid values: 1 - 999
Workers per node	Maximum number of worker threads involved on the system for any given node for each query. For example, on Teradata, the maximum number of AMPs that can be involved in each query on a node. The workers per node value should be determined based on the size of the system and how many AMPs there are per node. Valid values: 1 - 999
Memory needed per node	This displays a recommended amount of memory that you should reserve on each node in the system for Teradata QueryGrid operations. **Memory needed per node** is a calculated value based on the query concurrency value and the workers per node value supplied by the user. Depending on the available memory on the nodes certain settings may need to be tuned, for example, FSG Cache, to ensure memory is available for Teradata QueryGrid.

Editing a System

You can edit any of the system properties.

1. From the **Systems** view, click ▼ next to a system and click **Edit**.

Nodes Tab

Use the **Nodes** tab on the **Systems** view of the **QueryGrid** portlet to view the details of existing nodes and to associate a node with the system.

In addition to adding a node, you can take the following row actions:

Row Actions

Delete allows you to delete the selected node.

View Node Details allows you to view the network interfaces, fabrics, and connectors that this node is associated with.

Adding Nodes Automatically

Nodes are added by installing the tdqg-node package on each node along with an auto-generated configuration file.

Enter the IP Addresses or hostnames of the nodes to add or select the associated system monitored by Viewpoint to auto-populate the node list.

Perform the install of the tdqg-node package and the auto-generated configuration file across all nodes using SSH. Do the following to add a node automatically:

1. From the **Systems** view, select the system that you want to add nodes to.
2. On the **Nodes** tab, click ✚ next to **Nodes**.
3. Click **Auto Install: Perform the install of the tdqg-node package and the auto-generated configuration file across all nodes using SSH.**
4. Do one of the following to select the nodes on which to install the tdqg-node software:

Option	Description
Select a system from the Associated Viewpoint system list.	The **Nodes** box is auto-populated with the names of the nodes that are available in the system that you selected. The tdqg-node software will be installed on these nodes.
Enter the IP addresses or host names of the nodes in the **Nodes** box.	Manually enter the IP addresses or host names of the nodes. The tdqg-node software will be installed on these nodes.

5. In the **Install concurrency** box, type an integer to set the number of nodes that can install simultaneously.

   Valid values range from 1 to 100. The default value is 10.
6. In the **SSH user** box, type the name of the SSH user.

   The SSH user should have sudo privilege to install the RPM.
7. Under **Authentication Mechanism**, select **Password** or **SSH Key**, as appropriate, and type the corresponding value in the box.

   Choose the authentication mechanism that you normally use to log onto the nodes when using SSH.
8. Click **Save**.

   [Optional] You may **Abort** on **Node Installation Status**.

Adding Nodes Manually

Do the following to add a node to QueryGrid manually:

1. From the **Systems** view, select the system that you want to add nodes to.
2. On the **Nodes** tab, click ✚ next to **Nodes**.

3. Click **Manual Install: Download the auto-generated configuration file that needs to be saved on each node to complete the install.**
4. Select the number of days before the access token (in the generated file) will expire. The default is 7 days.

   The access token is the number of days that registration with the system remains open. Any installs completed after that time will not be added to the system.
5. Click **Generate File**.
   When finished, a link to the configuration file appears next to **Generated file:**, which you can use to download the file through the browser.
6. Click **Close**.

Postrequisite

To complete a manual installation, you need to perform the following tasks:

- Install the tdqg-node-*release.architecture*.rpm software package on each node in the system.
- Copy the generated configuration file to `/etc/opt/teradata/tdqg/node/tdqg-node.json` on each node in the system.

Viewing Node Details

View node details to see the network interfaces, fabrics, and connectors that the node is part of.

1. From the **Nodes** tab of the **Systems** view, click ▼ next to the node that you want and select **View Node Details**.

Data Centers View

Data centers are logical names for the physical locations of systems. With this information, Teradata QueryGrid can determine whether the communication between two systems is occurring across a LAN or WAN and can ensure communication with Teradata QueryGrid manager remains LAN local if a Teradata QueryGrid manager is available locally.

You can view the data center names and descriptions. You can sort on the column headers to find the details you want.

▼ **Table Actions**

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

▼ **Row Actions**

Edit allows you to change the name and description of a data center.

Delete allows you to delete a data center.

Adding a Data Center

Data centers provide logical names for the physical locations of systems. With this information, Teradata QueryGrid can determine whether the communication between two systems is occurring across a LAN or WAN.

1. From the **Data Centers** view, click next to **Data Centers**.
2. Type a name, up to 256 characters.
3. Type a description, up to 1000 characters.
4. Click **Save**.

Editing a Data Center

Edit a data center when you need to change its name or description.

1. From the **Data Centers** view, click next to the data center that you want to edit and select **Edit**.

Systems Tab

Use the **Systems** tab in the **Data Centers** view of the **QueryGrid** portlet to view the systems associated with the selected data center.

Managers Tab

Use the **Managers** tab in the **Data Centers** view of the **QueryGrid** portlet to view the Managers associated with the selected data center.

You can view the details of managers and sort on the column headers to find the details you want.

Table Actions

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Networks View

The **Networks** view displays details about the networks that are configured, and allows you to add, edit, and activate them.

Networks are defined by rules that map physical network interfaces to logical network definitions, either by interface name or by CIDR notation. Links refer to networks to determine which interfaces to use for data transfer. The rules are checked in the order in which they appear in the matching rules list.

▾ **Table Actions**

Clear Filters removes any content in the filter boxes.

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

The row actions that are available on each row vary depending on whether the network is a previous, pending, or active version.

▾ **Row Actions**

Activate allows you to change a pending or previous version of a network to the current (active) version.

Add Pending Version allows you to create a pending version of the selected network that you can edit and test.

Delete allows you to delete a network.

Edit allows you to change all configuration for a current version. If a pending version, you cannot change the name or description.

View allows you to view the details of a previous network version.

Adding a Network

1. From the **Networks** view, click + next to **Networks**.
2. Type a name, up to 256 characters.
3. Type a description, up to 1000 characters.
4. Select the format that you want to use for the matching rules, either **CIDR notation** or **Interface name**.

   You can select interface name and type in the interface name or you can select CIDR notation to use the Classless Inter-Domain Routing (CIDR) designation for the physical network interface instead of an interface name. You can type up to 500 characters.
5. [Optional] Click + next to **Matching rules** to add a rule to the list.

   You can click − to remove an existing rule.
6. Click **Save**.

Editing a Network

Edit an network when you need to change its name, description, or its rules.

1. From the **Networks** view, click ▾ next to a network and click **Edit**.

Links Tab

Use the **Links** tab in the **Networks** view of the **QueryGrid** portlet to view the links associated with the selected network. View details about the links, such as the initiating and target connectors.

Working with Active, Pending, or Previous Versions

Making changes to the Teradata QueryGrid configuration can potentially impact production workloads, so Teradata QueryGrid supports the ability to test configuration changes before those changes are activated. After the configuration change is tested and verified, it can be activated so that it becomes available to production workloads.

Changes to configuration entities (fabrics, connectors, links, communication policies, and networks) are versioned. When a query starts, it resolves which versions of the configuration entities it will use. These versions are passed around to all components executing on behalf of that query, so that all components use the same version of the configuration.

When a configuration change is made, it may take some time for that change to reach all the components of a system. If a query references a version of a configuration entity that has not been propagated throughout the system, the query will fail. Configuration changes should be persisted and tested before being applied to production workloads, to ensure that the configuration change is distributed to all the components of Teradata QueryGrid.

Configurations can be in one of three states: active, pending, and previous. Active is the tested and verified actively running version of the configuration. Pending is used for testing new or modified configurations. Active configurations are put in a previous state when another version of the configuration is activated.

If you replace an active version, the formerly active version becomes the previous version and the former previous version is deleted. If you activate a pending version, the formerly active version that you replaced becomes the previous version. If you activate a previous version, the active version becomes the previous version, and any pending version remains the pending version.

Replacing Active Network Versions with Pending or Previous Versions

1. Click **Networks**.
2. Click ▾ next to the pending or previous version that you want to make the active version and select **Activate**.

Communication Policies View

Communication policies define how systems communicate with one another. You can configure communication policies that have different amounts of concurrent streams used for transfer per request and levels of security to customize the connections between different Teradata QueryGrid systems. Communication policies are not necessarily specific to the systems involved, so you can reuse them.

The **QueryGrid** portlet is preconfigured with a policy appropriate to use over LANs (LAN Policy) and a policy appropriate for use with WANs (WAN Policy).

▾ **Table Actions**

Clear Filters removes any content in the filter boxes.

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

The row actions that are available on each row vary depending on whether the communication policy is a previous, pending, or active version.

Row Actions

Activate allows you to change a pending or previous version of a communication policy to the current (active) version.

Add Pending Version allows you to create a pending version of the selected communication policy that you can edit and test.

Delete allows you to delete a communication policy version. Deleting a current (active) version also deletes any pending or previous versions.

Edit allows you to change all communication policies configuration to the current (active) version. If communications policy is in the pending state, you cannot change the name or description.

View allows you to view the details of the previous communications policy version.

Adding a Communication Policy

1. From the **Communication Policies** view, click ✚ next to **Communication Policies**.
2. Type a policy name, up to 256 characters.
3. Type a description, up to 1000 characters.
4. Type an integer from 1 and 5 to set the number of data transfer streams that can run simultaneously. The default value is 1.
5. Select the combination of authentication, integrity and encryption checks that you want to use.

   A target connector is only as secure as the least secure communication policy referenced by a link referring to the target connector. The choices appear in order of increasing security level, with the choice that provides the greatest level of security listed last. Because the first choice provides only minimal security, it is recommended for use only in highly trusted environments. The second choice provides more security, but is still recommended for use only in a trusted environment. It is considered a best practice to use one of the more secure choices in non-trusted environments.

Option	Description
IP-based authentication, no integrity check, no Encryption	
IP-based authentication, checksum integrity check, no encryption	• Select the extent of the integrity checks to be run.
Key-based authentication, secure integrity check, encrypt credentials	• Select the extent of the integrity checks to be run. • Select the size of the authentication key that you want to use. By default, Teradata QueryGrid uses a 2048-bit authentication key. • Select the size of the encryption key that you want to use. By default, Teradata QueryGrid uses a 128-bit encryption key. • Select the encryption/integrity algorithm that you want to use. By default, Teradata QueryGrid uses AES-GCM. The choices appear in order of increasing security level, with the choice that provides the greatest level of security at the bottom.

Option	Description
Key-based authentication, secure integrity check, encrypt all data	• Select the extent of the integrity checks to be run. • Select the size of the authentication key that you want to use. By default, Teradata QueryGrid uses a 2048-bit authentication key. • Select the size of the encryption key that you want to use. By default, Teradata QueryGrid uses a 128-bit encryption key. • Select the encryption/integrity algorithm that you want to use. By default, Teradata QueryGrid uses AES-GCM. The choices appear in order of increasing security level, with the choice that provides the greatest level of security at the bottom.

6. Click **Save**.

Editing a Communication Policy

Edit a communication policy when you need to change its properties. You cannot change a policy's name or description if the policy is in pending state.

You cannot edit a previous version.

1. From the **Communication Policies** view, click ▾ next to the communication policy that you want to edit and select **Edit**.

Working with Active, Pending, or Previous Versions

Making changes to the Teradata QueryGrid configuration can potentially impact production workloads, so Teradata QueryGrid supports the ability to test configuration changes before those changes are activated. After the configuration change is tested and verified, it can be activated so that it becomes available to production workloads.

Changes to configuration entities (fabrics, connectors, links, communication policies, and networks) are versioned. When a query starts, it resolves which versions of the configuration entities it will use. These versions are passed around to all components executing on behalf of that query, so that all components use the same version of the configuration.

When a configuration change is made, it may take some time for that change to reach all the components of a system. If a query references a version of a configuration entity that has not been propagated throughout the system, the query will fail. Configuration changes should be persisted and tested before being applied to production workloads, to ensure that the configuration change is distributed to all the components of Teradata QueryGrid.

Configurations can be in one of three states: active, pending, and previous. Active is the tested and verified actively running version of the configuration. Pending is used for testing new or modified configurations. Active configurations are put in a previous state when another version of the configuration is activated.

If you replace an active version, the formerly active version becomes the previous version and the former previous version is deleted. If you activate a pending version, the formerly active version that you replaced becomes the previous version. If you activate a previous version, the active version becomes the previous version, and any pending version remains the pending version.

Replacing Active Communication Policies Versions with Pending or Previous Versions

1. Click **Communication Policies**.
2. Click ▾ next to the pending or previous version that you want to make the active version and select **Activate**.

User Mappings View

The **User Mappings** view allows you to construct a mapping table by mapping a username on one data source to a username on another data source. Using this feature, a user logged into one system can submit queries as a specific user on the remote data source. You can map multiple users to a single remote user if desired (note, you cannot map a single local user to multiple remote users). User mappings are associated with links.

You can view the mapping details. You can sort on the column headers to find the details you want.

▾ **Table Actions**

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

▾ **Row Actions**

Delete allows you to delete a user mapping.

Edit allows you to change the name and description for a user mapping.

Adding a User Mapping

1. From the **User Mappings** view, click + next to **User Mappings**.
2. Type a name for the mapping, up to 256 characters.
3. Type a description, up to 1000 characters.
4. Click **Save**.
5. In the list of user mappings, select the mapping that you just added.
6. On the **Users** tab, click + next to **Users**.
7. Type the name of the user on the initiating system.

   You can map multiple users to single remote user if desired, but you may use an initiating user only once within a mapping.
8. Type the name of the user on the target system that you want to map to.
9. Click **Save**.

Editing a User Mapping

Edit a user mapping when you need to change any of the user mapping properties.

1. From the **User Mappings** view, click ▾ next to the user mapping that you want to edit and select **Edit**.

Managers View

Teradata QueryGrid managers provide centralized management and monitoring of Teradata QueryGrid. Managers are used to install and upgrade software on Teradata QueryGrid systems. You can configure settings for managers, such as the length of time to retain log data and the session timeout.

Managers Tab

Use the **Managers** tab in the **Managers** view of the **QueryGrid** portlet to view details about existing managers.

You can view the details of managers and sort on the column headers to find the details you want.

Table Actions

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Row Actions

Delete allows you to delete a Teradata QueryGrid manager with a status of offline.

Service Accounts Tab

Service Accounts are used for Viewpoint and Teradata Support to access Teradata QueryGrid manager.

Table Actions

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Row Actions

Delete allows you to delete a service account.

Edit allows you to edit the password and description for a service account.

Adding a Service Account

To add a service account to Teradata QueryGrid manager, do the following:

1. In the **Managers** view, click the **Service Accounts** tab.
2. Click + next to **Service Accounts**.
3. Type a name in the **Username** box, up to 256 characters.
4. Type a password in the **Password** box, up to 256 characters.
5. Type the same password in the **Confirm password** box.

6. [Optional] Type a description in the **Description** box, up to 1000 characters.
7. Click **Save**.

Editing a Service Account

You cannot edit the Username property.

1. From the **Service Accounts** tab of the **Managers** view, click ▾ next to a service account and click **Edit**.

Resetting a Service Account Password

Use a command line tool `reset-password.sh` to reset a service account password on the Teradata QueryGrid manager system.

1. From the command line of the Teradata QueryGrid manager system, run: `/opt/teradata/tdqgm/bin/reset-password.sh`
   The user must have sudo privilege or be a member of the tdqgm group to run this command.

Settings Tab

You can edit some settings, such as the number of days to retain log data.

Editing Manager Settings

1. From the **Managers** view, click the **Settings** tab to edit the following settings for the managers:

Setting	Description
Log data retention	Number of days to retain log data. Valid values range from 1 to 365.
Query data retention	Number of days to retain query data. Valid values range from 1 to 365.
Query summary frequency	Number of seconds to wait between query summaries. Valid values range from 6 to 300.
Session timeout	Number of minutes to wait before a session times out. Valid values range from 10 to 1440.

2. Click **Apply**.

Software View

The **Software** view allows you to upload and delete connector, fabric, and node software. The **Software** view also displays details about the uploaded Teradata QueryGrid software packages. After uploading, the packages are available to install or upgrade.

The **Software** view includes the following tabs for uploading, deleting, and viewing details of the particular package:

- Connector Software
- Fabric Software

- Node Software

Connector Software Tab

Use the **Connector Software** tab in the **QueryGrid** portlet to view the details of existing connectors and to upload or delete connector software.

You can view software details and sort on the column headers to find the details you want.

Table Actions

Clear Filters removes any content in the filter boxes.

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Row Actions

Delete allows you to delete connector software.

Uploading Connector Software

Upload connector software when installing Teradata QueryGrid or when a new version of the connector software becomes available.

1. From the **Connector Software** tab of the **Software** view, click + next to **Connector Software**.
2. Browse to select the connector software.
3. Click **Upload**.
4. Click **OK** after the software has been successfully uploaded.

Fabric Software Tab

Use the **Fabric Software** tab in the **QueryGrid** portlet to view the details of existing fabric software and to upload or delete fabric software.

You can view fabric software details and sort on the column headers to find the details you want.

Table Actions

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Row Actions

Delete allows you to delete fabric software.

Uploading Fabric Software

Upload fabric software when installing Teradata QueryGrid or when a new version of the fabric software becomes available.

1. From the **Fabric Software** tab of the **Software** view, click ✚ next to **Fabric Software**.
2. Browse to select a fabric software.
3. Click **Upload**.
4. Click **OK** after the software has been successfully uploaded.

Node Software Tab

Use the **Node Software** tab in the **QueryGrid** portlet to view the details of existing node software and to upload or delete node software.

You can view software details and sort on the column headers to find the details you want.

Table Actions

Configure Columns allows you to choose the columns to display and set thresholds.

Export creates a .csv file containing all available data up to one million rows. If more than one million rows of data are available, then use the filters to display the data you want before exporting it.

For more information, see Summary Table Controls.

Row Actions

Delete allows you to delete node software.

Uploading Node Software

Upload node software when installing Teradata QueryGrid or when a new version of the node software becomes available.

1. From the **Node Software** tab of the **Software** view, click ✚ next to **Node Software**.
2. Browse to select the node software.
3. Click **Upload**.
4. Click **OK** after the software has been successfully uploaded.

APPENDIX B

QueryGrid Portlet Metrics

Communication Policies Metrics

Metric	Description
Name	Name given to this communication policy
Description	Description of this communication policy
Last Modified	Creation date or the last time this communication policy was edited

Connector Diagnostic Check Metrics

Metric	Description
Host Name	Host name of the connector
Status	Status of the connector: • Success • Failed • Waiting
Start Time	Time the diagnostic check started
End Time	Time the diagnostic check ended
Error Message	Text of the error message that was returned, if a problem was detected

Connector Software Metrics

Metric	Description
Name	Name of the connector software
Version	Version of the connector software
Platform	Platform that the connector software runs on
Package Date	Date that the connector software package was built
Package Size	Size of the connector software package

Connectors Metrics

Metric	Description
Name	Name of this connector
Description	Description that distinguishes this connector
Software	Version of software that this connector runs
Enabled	Whether this connector is enabled or disabled
System	System that this connector runs in
Nodes Online	Number of nodes on which the connector software is installed and both the node and fabric are available
Nodes Offline	Number of nodes on which the connector software is installed and not currently active
Drivers Online	Number of nodes on which the connector is installed and the driver is running
Drivers Offline	Number of nodes on which the connector driver is installed and not currently active but is supposed to be active
Fabric	Fabric that this connector runs in
Last Modified	Creation date or the last time this connector was edited

Data Centers Metrics

Metric	Description
Name	Name given to this data center
Description	Description of this data center

Fabrics Metrics

Metric	Description
Name	Name given to this fabric
Description	Description of this fabric
Port	Port on which this fabric runs
Fabric Software	Version of software that this fabric runs
Nodes Online	Number of nodes that are online for the fabric
Nodes Offline	Number of nodes that are supposed to be online for the fabric but are not
Last Modified	Creation date, or the last time this fabric was edited

Fabric Software Metrics

Metric	Description
Name	Name of the fabric software
Version	Version of the fabric software
Platform	Platform that the fabric software runs on
Package Date	Date that the fabric software package was built
Package Size	Size of the fabric software package

Issues History Metrics

Metric	Description
ID	Identification number assigned to the issue
Time	Time the issue was created, updated, or resolved depending on the state
Issue	Name of the issue
Severity	Critical or Warning
Type	Manager or System
Component	Name of the affected component
State	Specifies if the issue was created, updated, or deleted.

Issues Metrics

Metric	Description
All	Total number of issues
Critical	Number of issues considered to be critical
Warning	Number of warning-level issues
Issue	Description of the issue
Severity	Critical or Warning
Type	Origin of the issue, either in a manager or a system
Component	Name of the manager or system from which the issue originated
Time	Time the issue occurred
Duration	Length of time the issue lasted

Link Bandwith Test Metrics

Metric	Description
Host Name	Host name of the link
Status	Status of the link, either Success, Failed, or Waiting
Start Time	Time the link bandwidth test started
End Time	Time the link bandwidth test ended
Error Message	Text of the error message that was returned, if a problem was detected

Links Metrics

Metric	Description
Name	Name of this link
Description	Description that distinguishes this link
Initiating Connector	Connector that initiates and sends a query to a target system
Initiating System	System on which a query originates
Target Connector	Connector that receives and processes a query from the initiating system or accepts data from the initiating system, and may return results to the initiating system
Target System	System on which a query executes
Communication Policy	Communication policy that this link uses
Fabric	Fabric that this link runs in
Last Modified	Creation date or the last time this connector was edited

Managers Metrics

Metric	Description
Host Name	Name given to this manager
Status	Online or Offline
Version	Software release that runs on this manager
Data Center	Data center where this manager is located
Public Address	Public-facing host name or IP address that is used to communicate with this manager
Free Space	The amount of free disk space available on the manager.
Disk Usage %	Percentage of the disk used by this manager

Table continued on next page.

Metric	Description
Last Heartbeat	Last time this manager sent a heartbeat

Networks Metrics

Metric	Description
Name	Name given to the network
Description	Description of this network
Last Modified	Creation time or last time this network was edited

Node Details Metrics

Node Details

Metric	Description
Node Name	Name given to this node
Platform	Platform that this node runs on
Node Software	Version of software that this node runs
Last Heartbeat	Last time this node sent a heartbeat
Status	Whether this node is currently online or offline

Network Interfaces

Metric	Description
Interface Name	Name of a network interface that this node is a part of
Address	Address of the network interface

Fabrics

Metric	Description
Fabric Name	Name of a fabric that this node is a part of
Fabric Software	Version of software that runs on the fabric
Process ID	Identification number given to the fabric process
Status	Whether the fabric is currently online or offline
Drivername	Name of the driver
Status	Whether the driver in the fabric is currently online or offline
Process ID	Identification number of the driver process in this fabric

Connectors

Metric	Description
Connector Name	Name of a connector that this node is a part of
Connector Software	Version of software that runs on the connector
Status	Whether the connector is currently online or offline
Fabric Name	Name of the fabric that the connector is part of

Node Installation Status Metrics

Metric	Description
All	Total number of nodes
Success	You can filter to see the number of nodes in the following states: • Success: Number of node installations that have succeeded • Failed: Number of node installations that have failed • Installing: Number of nodes that are currently installing • Pending: Number of nodes that are currently pending • Aborted: Number of node installations that have aborted
Host Name	Host name of the node
Status	Status is: Success, Failed, Installing, Pending, or Aborted
Start Time	Time that installation of the node started
End Time	Time that installation of the node ended
Error Message	Error message that resulted from installation, if any

Node Software Metrics

Metric	Description
Name	Name of the node software package
Version	Version of the node software package
Platform	Platform that the of the node software package runs on
Package Date	Date when the package was built
Package Size	Size of the software package

Node Status Metrics

Metric	Description
All	Total number of nodes in the fabric
Online	Number of nodes that are currently active
Offline	Number of nodes that are currently not active
Host Name	Host name of the device that is the node
Status	Whether the node is currently online or offline
System	System in which the node runs

Service Account Metrics

Metric	Description
User Name	Name given to the service account
Description	Description of the service account

System Metrics

Metric	Description
Name	Name of the system
Description	Description of the system
Data Center	Data center that the system is a part of
Node Software	Version of node software that the nodes in the system run
Nodes Online	Number of nodes that are currently active on the system
Nodes Offline	Number of nodes that are not currently active on the system

System Nodes Metrics

Metric	Description
Host Name	Host name of the node
Status	Whether the node is online or offline
Platform	Platform on which the node runs
Node Software	Version of node software that the node runs
Last Heartbeat	Last time the node sent a heartbeat

User Mappings Metrics

Metric	Description
Group Name	Name of the user mapping group that the user belongs to
Description	Description of the user mapping group
Mapping Count	Number of mapping relationships created for this user mapping group
Last Modified	Time that the user mapping group was last changed

User Metrics

Metric	Description
Initiating User	Name of the user on the system where the query originates
Target User	Name of the user on the system where the query executes

APPENDIX C

QueryGrid Data Transformation

Data Type Mapping for Teradata QueryGrid Connectors

When you query a remote system that runs a connector that is a different platform from the local system connector, data is converted to satisfy the target database requirements. Even when the remote and local connectors are the same, the data is still transformed because the host formats might be different from each other.

It is important to note the following information about data types:

- The mapping for some data types may not preserve the data in all cases, so be aware that with some queries, data loss can occur.

  As an example, the import of data from Presto with an unlimited VARCHAR size into a limited Teradata VARCHAR column requires truncation and thus, data loss.

  As another example, if you export a Teradata byteint data type as a Boolean and then import it again, the data may not be the same. This can occur because the Boolean zero or one is imported while the byteint may have contained other values beyond zero or one.
- If a type is unsupported on the remote system, the remote connector returns an error message for that column to the local system. For `SELECT *` queries with a column containing an unsupported data type, the unsupported column causes the entire query to return an error. In this case the user must query the supported columns individually.
- All time-related data types are in UTC.

Global Data Map

The global data map is used to transform data from the data format of the remote system to the data format of the local system. Data fetched from the remote system is mapped into a global format, then the global data is mapped to the format of the local system. To achieve this every connector has two mappings, connector to global and global to connector; for example, Teradata to global, global to Teradata, Presto to global, global to Presto, and so on for each connector type.

The following global data types are supported:

Supported Global Data Types
Integer Data Types
G_Integer
G_BigInt
G_ByteInt

Table continued on next page.

Supported Global Data Types
G_SmallInt
G_Number
Float Data Types
G_Decimal
G_Double
G_Float
Time Data Types
G_Time
G_TimeWithTimeZone
G_TimeStamp
G_TimeStampWithTimeZone
G_Dat
Binary Data Types
G_Byte
G_Varbyte
G_Blob
G_Boolean
Character Data Types
G_Char_Latin
G_Varchar_UTF8
G_Varchar_UTF16
G_Clob
G_LongVarchar
Complex Data Types
G_CharSet
G_Vargraphic
G_LongVargraphic
G_XML
G_JSON
G_Array
G_Map

Table continued on next page.

Supported Global Data Types
G_Row
Intervals Data Types
G_IntervalYear
G_IntervalYearToMonth
G_IntervalDayToSecond
G_IntervalDay
G_IntervalDayToHour
G_IntervalDayToMinute
G_IntervalHour
G_IntervalHourToMinute
G_IntervalHourToSecond
G_IntervalMinute
G_IntervalMinuteToSecond
G_IntervalSecond
G_IntervalMonth
G_Period
Others
G_Varchar_Latin
G_Char_UTF8
G_Char_UTF16
G_Array_VC
G_Struct
G_Struct_VC
G_Bit
G_Varbit
G_LongVarbit

Example: Teradata to Presto Data Transformation

The following table is shown as an example of a data transformation. The table shows a mapping of Teradata data types to Presto data types. For example, an insert from Teradata into Presto converts Teradata data types into Presto data types. The table does not show the interim transformations between Teradata to the global map and the global mapping to Presto.

Teradata Data Type	Presto Data Type
INTEGER	INTEGER
TINYINT	TINYINT
BIGINT	BIGINT
BYTEINT	TINYINT
SMALLINT	SMALLINT
DECIMAL	DECIMAL
DOUBLE	DOUBLE
FLOAT	DOUBLE
REAL	DOUBLE
TIME	TIME
TIME with TimeZone	TIME with TimeZone
TIMESTAMP	TIMESTAMP
TIMESTAMP with TimeZone	TIMESTAMP with TimeZone
DATE	DATE
BYTE	VARBINARY
VARBYTE	VARBINARY
CHAR(X)	CHAR(X) **Note:** This is CHARACTER UTF8 on Presto.
VARCHAR(X)	VARCHAR(X) **Note:** This is CHARACTER UTF8 on Presto.
VARCHAR UTF16	VARCHAR(X) **Note:** This is CHARACTER UTF8 on Presto.
VARGRAPHIC	VARCHAR UTF8
IntervalYear	VARCHAR UTF8
IntervalYearToMonth	VARCHAR UTF8
IntervalDayToSecond	VARCHAR UTF8

Table continued on next page.

Teradata Data Type	Presto Data Type
IntervalDay	VARCHAR UTF8
IntervalDayToHour	VARCHAR UTF8
IntervalDayToMinute	VARCHAR UTF8
IntervalHour	VARCHAR UTF8
IntervalHourToMinute	VARCHAR UTF8
IntervalHourToSecond	VARCHAR UTF8
IntervalMinute	VARCHAR UTF8
IntervalMinuteToSecond	VARCHAR UTF8
IntervalSecond	VARCHAR UTF8
IntervalMonth	VARCHAR UTF8

The following types cannot be mapped from Teradata to Presto:

- ARRAY
- BLOB
- CLOB
- Geospatial
- JSON
- Period
- Distinct UDT
- Structured UDT
- XML

Example: Presto to Teradata Data Transformation

The following table is shown as an example of a data transformation. The following table maps Presto data types to Teradata data types. For example, a select from Presto into Teradata converts Presto data types into Teradata data types. The table does not show the interim transformations between Presto to the global map and the global mapping to Teradata.

Presto Data Type	Teradata Data Type
BOOLEAN	BYTEINT
INTEGER	INTEGER
SMALLINT	SMALLINT
TINYINT	TINYINT
BIGINT	BIGNIT
FLOAT	FLOAT

Table continued on next page.

Presto Data Type	Teradata Data Type
DOUBLE	FLOAT
DECIMAL	DECIMAL
CHAR(X)	CHAR(X) **Note:** By default, Presto CHAR(X) is mapped to character set UNICODE.
VARCHAR(X)	VARCHAR(X)
VARBINARY	VARBINARY
DATE	DATE
TIME with TimeZone	TIME with TimeZone
TIMESTAMP	TIMESTAMP
TIMESTAMP with TimeZone	TIMESTAMP with TimeZone
IntervalYearToMonth	IntervalYearToMonth
IntervalDayToSecond	IntervalDayToSecond
ARRAY	VARCHAR(X) CHARACTER SET UNICODE
MAP	VARCHAR(X) CHARACTER SET UNICODE
ROW	VARCHAR(X) CHARACTER SET UNICODE

The following types cannot be mapped from Presto to Teradata:

- ARRAY
- JSON
- MAP
- ROW
- TIME

Data Transformation Between Similar Hosts

Presto to Presto

Although Presto to Presto connections are between similar hosts, data transformation still occurs. The following table shows the data types that are transformed from the remote Presto system to the local Presto system and are a different format after the transformation.

Presto Data Type (remote)	Presto Data Type (local)
Map(datatype1, datatype2)	Map(varchar,varchar)
ROW	VARCHAR

APPENDIX D
Notation Conventions

About Notation Conventions

This appendix describes the notation conventions used in this book.

Convention	Description
Syntax Diagrams	Describes SQL syntax form, including options.
Square braces in the text	Represent options. The indicated parentheses are required when you specify options. For example: DECIMAL [(n[,m])] means the decimal data type can be defined optionally: • without specifying the precision value n or scale value m • specifying precision (n) only • specifying both values (n,m) You cannot specify scale without first defining precision. • CHARACTER [(n)] means that use of (n) is optional. The values for n and m are integers in all cases.

Syntax Diagram Conventions

Notation Conventions

Item	Definition and Comments
Letter	An uppercase or lowercase alphabetic character ranging from A through Z.
Number	A digit ranging from 0 through 9. Do not use commas when typing a number with more than 3 digits.
Word	Keywords and variables. • UPPERCASE LETTERS represent a keyword. Syntax diagrams show all keywords in uppercase, unless operating system restrictions require them to be in lowercase. • lowercase letters represent a keyword that you must type in lowercase, such as a Linux command.

Table continued on next page.

Item	Definition and Comments
	• Mixed Case letters represent exceptions to uppercase and lowercase rules. The exceptions are noted in the syntax explanation. • *lowercase italic letters* represent a variable such as a column or table name. Substitute the variable with a proper value. • **lowercase bold letters** represent an excerpt from the diagram. The excerpt is defined immediately following the diagram that contains it. • <u>UNDERLINED LETTERS</u> represent the default value. This applies to both uppercase and lowercase words.
Spaces	Use one space between items such as keywords or variables.
Punctuation	Type all punctuation exactly as it appears in the diagram.

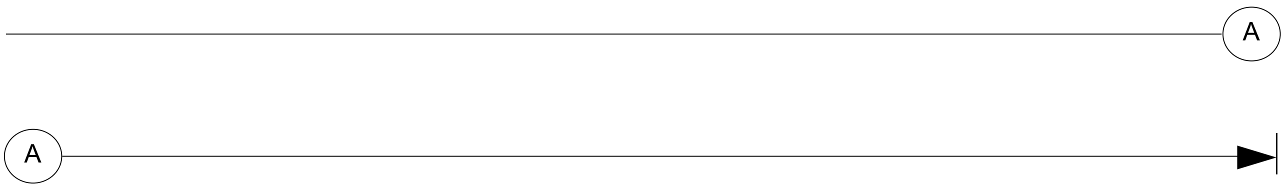
Paths

The main path along the syntax diagram begins at the left with a keyword, and proceeds, left to right, to the vertical bar, which marks the end of the diagram. Paths that do not have an arrow or a vertical bar only show portions of the syntax.

The only part of a path that reads from right to left is a loop.

Continuation Links

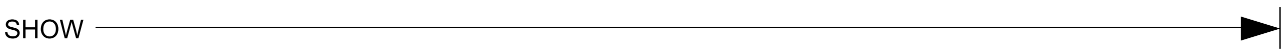
Paths that are too long for one line use continuation links. Continuation links are circled letters indicating the beginning and end of a link:



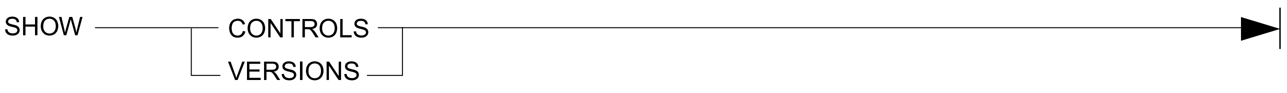
When you see a circled letter in a syntax diagram, go to the corresponding circled letter and continue reading.

Required Entries

Required entries appear on the main path:

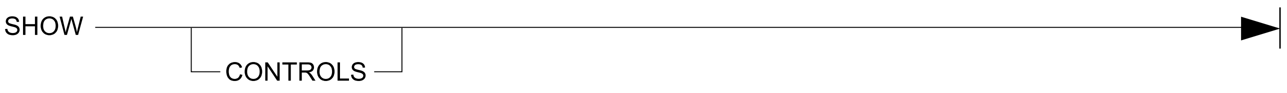


If you can choose from more than one entry, the choices appear vertically, in a stack. The first entry appears on the main path:

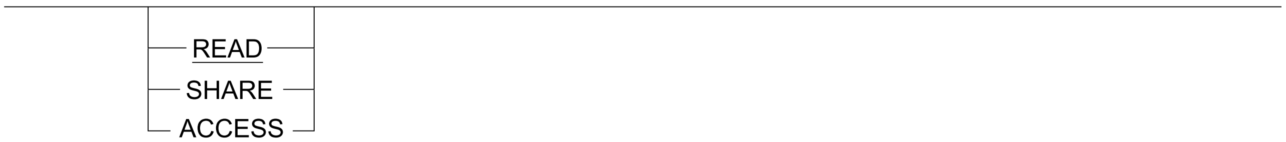


Optional Entries

You may choose to include or disregard optional entries. Optional entries appear below the main path:



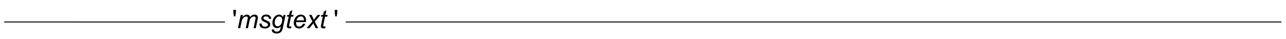
If you can optionally choose from more than one entry, all the choices appear below the main path:



Some commands and statements treat one of the optional choices as a default value. This value is UNDERLINED. It is presumed to be selected if you type the command or statement without specifying one of the options.

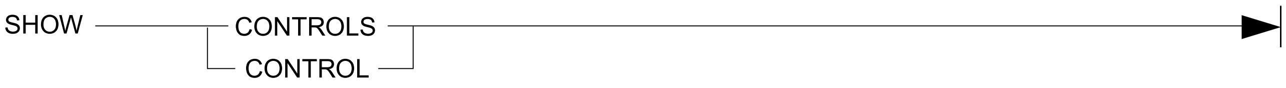
Strings

String literals appear in apostrophes:



Abbreviations

If a keyword or a reserved word has a valid abbreviation, the unabbreviated form always appears on the main path. The shortest valid abbreviation appears beneath.

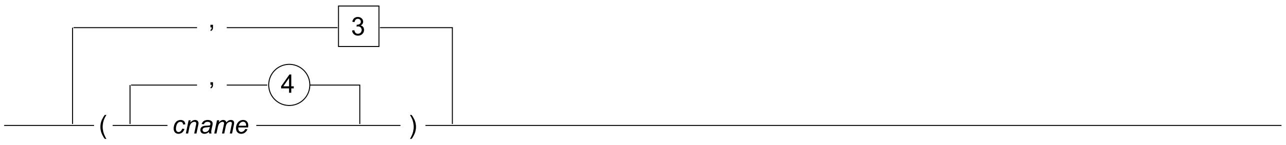


In the above syntax, the following formats are valid:

SHOW CONTROLS
SHOW CONTROL

Loops

A loop is an entry or a group of entries that you can repeat one or more times. Syntax diagrams show loops as a return path above the main path, over the item or items that you can repeat:



Read loops from right to left.

The following conventions apply to loops:

Item	Description	Example
maximum number of entries allowed	The number appears in a circle on the return path.	In the example, you may type *cname* a maximum of four times.
minimum number of entries allowed	The number appears in a square on the return path.	In the example, you must type at least three groups of column names.
separator character required between entries	The character appears on the return path. If the diagram does not show a separator character, use one blank space.	In the example, the separator character is a comma.

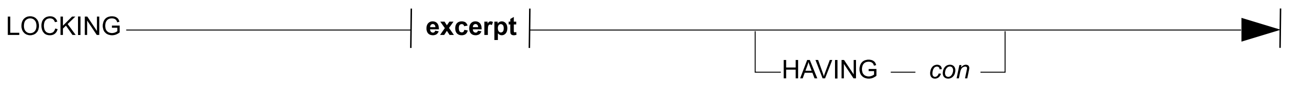
Table continued on next page.

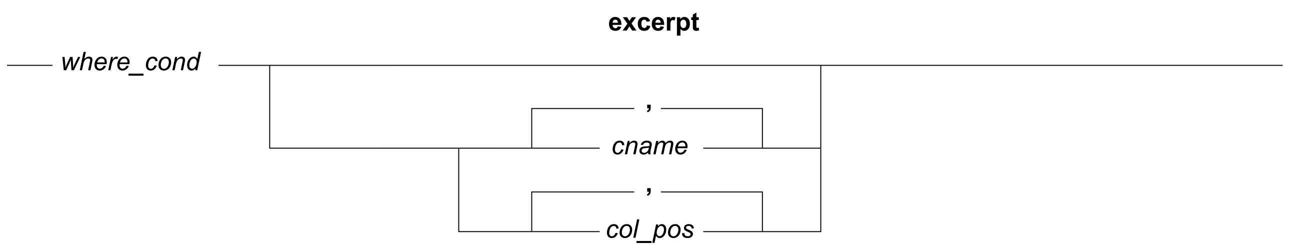
Item	Description	Example
delimiter character required around entries	The beginning and end characters appear outside the return path. Generally, a space is not needed between delimiter characters and entries.	In the example, the delimiter characters are the left and right parentheses.

Excerpts

Sometimes a piece of a syntax phrase is too large to fit into the diagram. Such a phrase is indicated by a break in the path, marked by (|) terminators on each side of the break. The name for the excerpted piece appears between the terminators in boldface type.

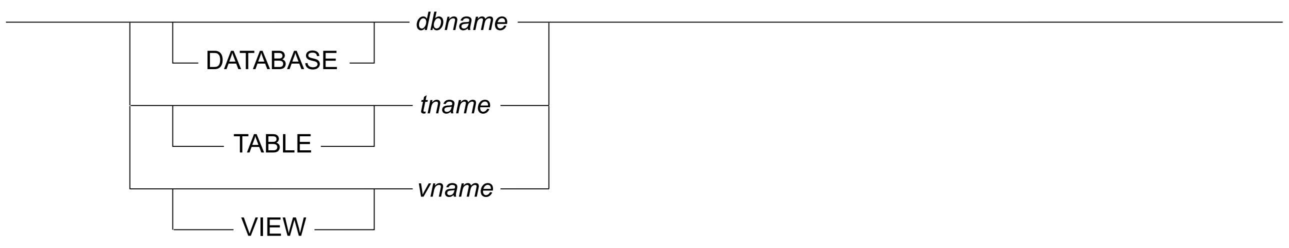
The boldface excerpt name and the excerpted phrase appears immediately after the main diagram. The excerpted phrase starts and ends with a plain horizontal line:





Multiple Legitimate Phrases

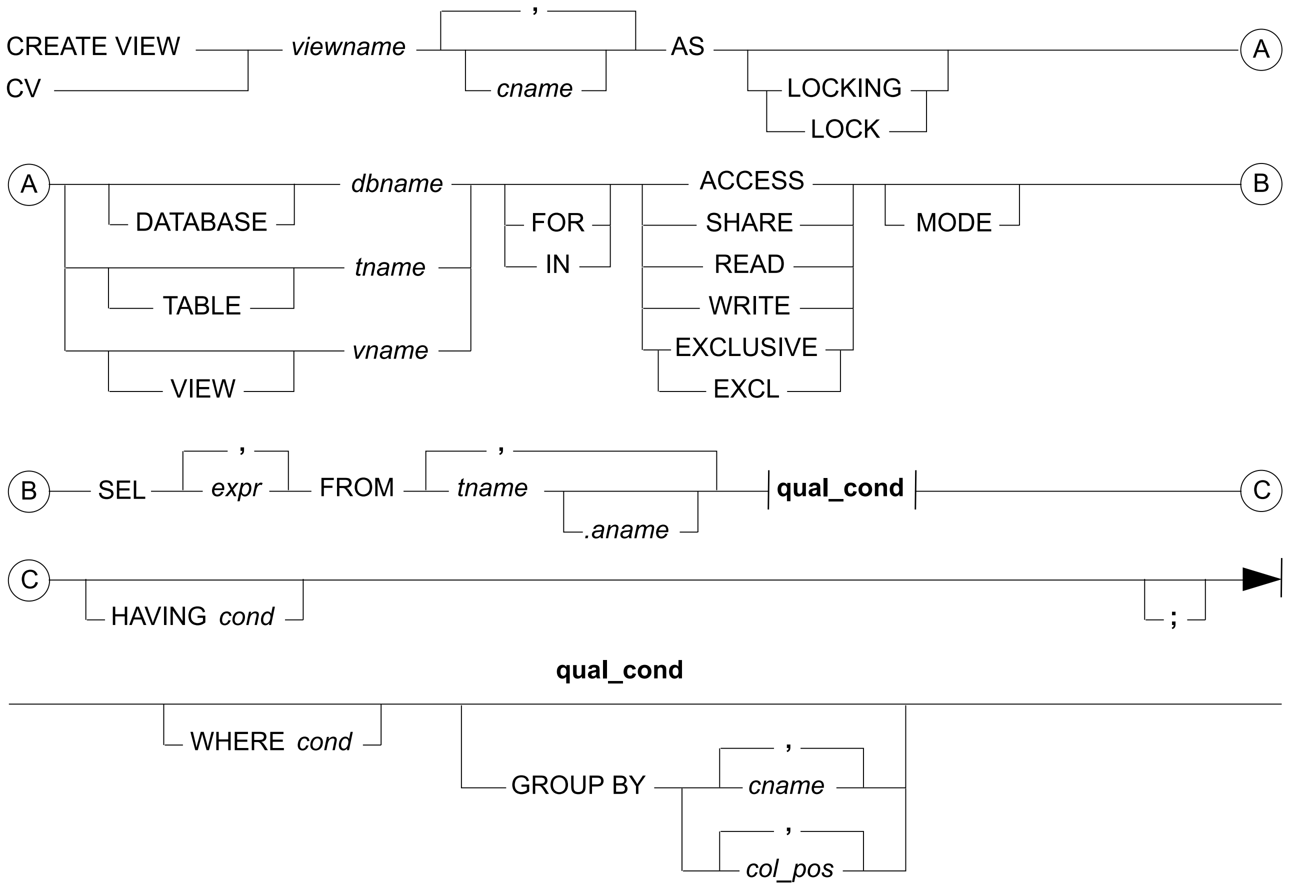
In a syntax diagram, it is possible for any number of phrases to be legitimate:



In this example, any of the following phrases are legitimate:

dbname
DATABASE *dbname*
tname
TABLE *tname*
vname
VIEW *vname*

Sample Syntax Diagram


CREATE VIEW
CV
viewname
,
cname
AS
LOCKING
LOCK
A
A
DATABASE
dbname
TABLE
tname
VIEW
vname
FOR
IN
ACCESS
SHARE
READ
WRITE
EXCLUSIVE
EXCL
MODE
B
B
SEL
,
expr
FROM
,
tname
.aname
qual_cond
C
C
HAVING cond
;
qual_cond
WHERE cond
GROUP BY
,
cname
,
col_pos