README - Teradata producer

The Teradata producer is a small stand-alone Java application that monitors activity on a Teradata system and sends events to the Enterprise Search server over a Kafka connection. The information it generates helps improve the quality of search results as it can track usage of data (such as tables and columns), which can be used to improve the ranking of search results provided by the Enterprise Search service. It also helps identify who used the database and adds this information to the search results.

The producer is designed to be run regularly (using a cron job or similar) at intervals of time that are set by the system administrator. The more often it is run, the more accurate the search results are likely to be. After the program is run, it creates a checkpoint file that contains the time stamp when it was last run. The next time the producer is run, the time stamp is used to identify new log entries, and the producer will only look for new entries in the DBQL query log.

Prerequisites

Java 1.8 must be installed on the system where the producer runs.

Download location

The producer is downloaded as part of the IBM Information Server suite installation, but must be configured individually. After installing Information Server, you can find the Teradata producer in the following directory:

```
<IS_install_path>/DatabaseSupport/GraphProducers/TERADATA-graphproducer
```

Installing and running the producer

To install, enable, and run the Teradata producer, complete these steps:

1. Make sure Teradata Database Query Log (DBQL) is enabled and working as described in *Enabling DBQL query logging*.
2. Copy the Teradata producer JAR file to the Teradata system.
3. Copy the associated property file to the system and edit as needed (see *Installing and configuring the producer*).
4. Test the producer from a terminal.
5. Optional: Set up a cron job to run the producer at regular intervals. For more information, see the reference page for the cron command on your system.

To complete these steps, you, as Database Administrator (DBA) user, must have access to the Teradata system and must have the required permissions to use DBQL.

Enabling DBQL query logging

The Teradata producer relies on certain Teradata activities being logged by the Teradata system. Enable this logging by following these steps:

1. Grant the DBA user permission to create access logging rules by running the following command:

   ```
   GRANT EXECUTE ON DBC.DBQLAccessMacro TO SYSDBA;
   ```

2. Check whether DBQL is already enabled by running the following command:

   ```
   SELECT * FROM DBC.DBQLRulesV;
   ```

3. Start query logging by running the following commands. Note that the DBA user can start or end collection for a specific user, a specific account, a group of users, or a list of accounts. Modify the command as required.

   ```
   BEGIN QUERY LOGGING WITH SQL, OBJECTS LIMIT SQLTEXT=0 ON ALL;
   BEGIN QUERY LOGGING WITH ALL ON <user>;
   ```

   This command will run from BTEQ only and not from any other client tools.

4. Check whether the rules were created and if there are entries in the DBQL.

   ```
   SELECT * FROM DBC.DBQLRulesV;SELECT * FROM DBC.QryLog ORDER BY
   STARTTIME DESC
   ```

5. To verify that events are being captured by Teradata, run some SQL queries against the database and then run the following query using the BTEQ utility.

   ```
   SELECT DISTINCT UserName, StartTime, QueryText, StepStartTime,
   StepStopTime FROM DBC.DBQLStepTbl S, DBC.DBQLogTbl L WHERE QueryText is
   not null AND S.queryid = L.queryid ORDER BY S.StepStopTime DESC;
   ```

   This query should return all data in those logs that are important to monitor. If the query does not return any results, run the following command to flush query logging data from memory to table:

   ```
   FLUSH QUERY LOGGING with ALLDBQL;
   ```

After query logging is enabled, install and configure the producer.

Installing and configuring the producer

The Teradata producer is provided as a stand-alone Java JAR file with an associated configuration file. These files should be installed on the Teradata system being monitored to get the best performance. However, you can also install and run the producer on another system. The configuration file should be named `bridge-agent.properties` . You can copy the `bridge-agent.properties.sample` file to a `bridge-agent.properties` file. The properties file and the JAR file should be colocated in the same directory.

A sample properties file is shown below. Most of the specified values should work for your system, but some specific ones must be changed. You must edit the `kafka.bootstrap.servers` property to contain the server name or IP address of the Information Server installation. You can probably leave the port number unchanged. The other Kafka settings (properties prefixed with `kafka`) in the sample should work on your system. However, the Kafka server and topic settings must match the Kafka consumer settings.

For Teradata settings (properties prefixed with `teradata`), you must update the following ones:

- The `teradata.url` property must contain the server name of the Teradata system to be monitored.
- The `teradata.userid` and `teradata.password` properties must contain the credentials of a user with sufficient access rights to allow the producer to access and retrieve the DBQL query log (such as the DBA user).

The checkpoint file defined in `teradata.bridge.checkpointfile` is used to store the time stamp of the last query processed so that each time the agent runs it only looks for new activity. To reprocess all entries, remove the checkpoint file or delete all its entries. The default file name is `connector.chkpt` but you can change the name as required.

If you use the commands exactly as described before, you do not need to change any of the `teradata.activityMonitor` properties.

```
kafka.bootstrap.servers=<serverwherekafkaisrunning>:59092
kafka.producer.topic.id=igc-kg-bridge-out
kafka.key.serializer=org.apache.kafka.common.serialization.StringSerializer
kafka.value.serializer=org.apache.kafka.common.serialization.StringSerializer
kafka.acks=all
kafka.retries=0
bridge.base.property=teradata

teradata.url=jdbc:teradata://localhost
teradata.userid=<userid>
teradata.password=<password>

teradata.activityMonitor.query=select distinct UserName, StartTime,
QueryText, StepStartTime, StepStopTime, DefaultDatabase from DBC.DBQLStepTbl
S, DBC.DBQLogTbl L where QueryText is not null AND S.queryid = L.queryid and
S.StepStopTime > CAST(? AS TIMESTAMP) order by S.StepStopTime desc;
teradata.bridge.checkpointfile=connector.chkpt
```

Running the producer

Start the Teradata producer by using the following command:

```
java -jar bridge-teradata-0.1.0.jar <Filepath to bridge-agent.properties>
```