Progress DataDirect for JDBC for
Google Analytics User's Guide

Release 6.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for JDBC for Google Analytics Driver

The Progress® DataDirect® for JDBC™ for Google Analytics™ driver supports SQL read-only access for JDBC applications to Google Analytics. The driver supports Google Analytics Management API version 3. In addition, the driver employs a SQL engine component that provides support to SQL constructs to enable the most comprehensive SQL support and JDBC standard connectivity. To support SQL access to Google Analytics services, the driver creates a relational map of the Google Analytics data model and translates SQL statements to Google Analytics REST API requests.

For an overview of how the driver exposes the Google Analytics data model as a relational schema, and how to query Google Analytics, see Introduction to the Google Analytics Data Model.

The documentation for the driver also includes the *Progress DataDirect for JDBC Drivers Reference*. The reference provides general reference information for all DataDirect drivers for JDBC, including content on troubleshooting, supported SQL escapes, and DataDirect tools.

For the complete documentation set, visit the Progress DataDirect Connectors Documentation Hub: https://docs.progress.com/category/datadirect-google-analytics.

For details, see the following topics:

- What's new in this release?
- Requirements
- Installing and setting up the driver
- Driver and DataSource classes
- Connection URL examples
- Data types

- SQL escape sequences
- DataDirect tools
- Troubleshooting
- Contacting Technical Support

What's new in this release?

Highlights of 6.0.0 Release

- The driver supports SQL read-only access to Google Analytics. See Supported SQL statements and extensions on page 79 for details.
- The driver supports JDBC core functions. For details, refer to "JDBC support" in the *Progress DataDirect for JDBC Drivers Reference*.
- The driver supports Google Analytics data types through data type inference. See Data types on page 15 and getTypeInfo() on page 15 for details.
- The driver supports OAuth 2.0 authentication. See OAuth 2.0 authentication on page 40 for further details.
- The driver supports the handling of large result sets with paging, and the FetchSize on page 61 connection property.
- The driver includes the Progress DataDirect Google Analytics Configuration Manager for quick configuration and testing of your driver in a web browser. The tool allows you to:
  - Generate and edit connection URLs
  - Test connect your connection URLs
  - Add tables to a relational view of Google Analytics data
  - Execute SQL commands for testing

  For details, see Generating connection URLs with the Configuration Manager on page 33 and Testing connections and queries on page 35.

Requirements

The driver is compatible with JDBC 2.0, 3.0, and 4.0.

The driver requires a Java Virtual Machine (JVM) that is Java SE 8 or higher, including Oracle JDK, OpenJDK, and IBM SDK (Java) distributions.

Installing and setting up the driver

This section provides you with an overview of the steps required to install and set-up the driver. After completing this procedure, you will be able to begin accessing data with your application.

To begin accessing data with the driver:

1. Install the driver:

   a) After downloading the product, unzip the installer files to a temporary directory.

   b) From the installer directory, run the appropriate installer file to start the installer.

   - **Windows**: `PROGRESS_DATADIRECT_JDBC_INSTALL.exe`
   - **Non-Windows**: `PROGRESS_DATADIRECT_JDBC_INSTALL.jar`

   c) Follow the prompts to complete installation.

   The installer program supports multiple installation methods, including command-line and silent installations. For detailed instructions, refer to the *Progress DataDirect for JDBC Drivers Installation Guide*.

2. Set your system CLASSPATH to include the driver `.jar` file. The CLASSPATH is the search string your Java Virtual Machine (JVM) uses to locate JDBC drivers on your computer. The following examples demonstrate setting the CLASSPATH from a command line using the default installation directory.

   - **Windows Example**

     ```
     CLASSPATH=.;C:\Program Files\Progress\DataDirect\JDBC\lib\60\googleanalytics.jar
     ```

   - **UNIX/LINUX Example**

     ```
     CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/googleanalytics.jar
     ```

3. Configure your driver using one of the following methods:

   - **Connection URL**: You can begin using the driver immediately by passing a connection URL with your application or tool. The following example shows how to connect using the OAuth 2.0 refresh token grant.

     ```
     jdbc:datadirect:googleanalytics:ClientID=client_id;ClientSecret=client_secret;
     RefreshToken=refresh_token;
     ```

     Note: See OAuth 2.0 authentication on page 40 for details.

     You can also generate a connection string using the Progress DataDirect Google Analytics Configuration Manager. For details, see Generating connection URLs with the Configuration Manager on page 33.

   - **Data sources**: The driver also supports connecting using JDBC data sources. A JDBC data source is a Java object, specifically a DataSource object, that defines connection information required for a JDBC driver to connect to the database. See Connecting using data sources for more information.

   Note: For most connections, specifying the minimum required connection properties is sufficient to begin accessing data; however, you can provide values for optional properties to use additional supported features and improve performance.

4. Set the values for any optional properties that you want to configure. For additional information on optional features and functionality, see the following resources:

   - Connection URL examples provides connection string examples that can be used to configure common functionality and features. You can modify and combine these examples to create a string that best suites your environment.
   - Connection property descriptions provides a complete list of supported properties by functionality.

- Performance considerations describes connection properties that affect performance, along with recommended settings.

5. Connect to your service and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:

   - Progress DataDirect Google Analytics Configuration Manager: The Google Analytics Configuration Manager is a browser-based tool that allows you to quickly generate connection URLs, test connections, and execute test queries.
   - Tableau: Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
   - DbVisualizer: DB Visualizer is a database tool that allows you to connect and execute SQL statements against your data.
   - Interactive SQL for JDBC (JDBCISQL) JDBCISQL is a command line interface that allows you to connect to a data source, execute SQL statements and retrieve results for display on a terminal.
   - Supported SQL statements and extensions: This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

Driver and DataSource classes

The following are the `Driver` and `DataSource` classes used by the driver:

Driver class:

com.ddtek.jdbc.googleanalytics.GoogleAnalyticsDriver

DataSource class:

com.ddtek.jdbcx.googleanalytics.GoogleAnalyticsDataSource

Connection URL examples

After setting the CLASSPATH, the connection information needs to be passed in the form of a connection URL. This section provides examples of connection strings configured to use common features and functionality. You can modify and/or combine these examples to create a connection string for your environment.

Note:

- You can also use the DataDirect Configuration Manager tool to generate and test connection URLs. For more information, see "Generating connection URLs with the Configuration Manager."
- Connection property names are case-insensitive. For example, `AuthenticationMethod` is the same as `authenticationmethod`.
- For connection properties that support string values, use the following escape sequence to specify values containing leading or trailing spaces and curly brackets: {*value*}. For example: `ProxyPassword={hello }` or `ProxyPassword={{hello}}`.

See also

Generating connection URLs with the Configuration Manager on page 33

OAuth 2.0 access token example

You may configure the driver to access Google Analytics resources using OAuth 2.0. In the connection URL, you may provide an access token to connect. However, it should be noted that, in most cases, an access token is available for a short period of time and may only be used for a single session.

The following connection URL uses an access token.

```
jdbc:datadirect:googleanalytics:AccessToken=access_token;[property=value[;...]];
```

where:

access_token

specifies the access token required to authenticate to Google Analytics. This property allows you to set the access token manually.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The AccessToken and RefreshToken properties may both be specified. If a value for the AccessToken property is not specified, the driver uses the value of the RefreshToken property to make a connection. If both values are not specified, the driver cannot make a successful connection. If both are specified, the driver ignores the AccessToken value and uses the RefreshToken value to generate a new AccessToken value.

For information on setting up OAuth 2.0 authentication, including obtaining access and refresh tokens, see OAuth 2.0 authentication.

OAuth 2.0 refresh token grant

You may configure the driver to access Google Analytics resources using OAuth 2.0. In the connection URL, you may provide a refresh token to connect using the refresh token grant. With a valid refresh token, the driver can obtain a new access token. In this way, a refresh token enables application access to Google for multiple sessions over an extended period of time.

The following connection URL uses the refresh token grant.

```
jdbc:datadirect:googleanalytics:RefreshToken=refresh_token;[property=value[;...]];
```

where:

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The AccessToken and RefreshToken properties may both be specified. If a value for the AccessToken property is not specified, the driver uses the value of the RefreshToken property to make a connection. If both values are not specified, the driver cannot make a successful connection. If both are specified, the driver ignores the AccessToken value and uses the RefreshToken value to generate a new AccessToken value.

For information on setting up OAuth 2.0 authentication, including obtaining access and refresh tokens, see OAuth 2.0 authentication.

Proxy server

This string includes the properties you may need to connect through a proxy server with OAuth 2.0 refresh token grant authentication.

```
jdbc:datadirect:googleanalytics:ProxyHost=proxy_host;ProxyPassword=proxy_password;
ProxyPort=proxy_port;ProxyUser=proxy_user;RefreshToken=refresh_token;
[property=value[;...]];
```

where:

proxy_host

specifies the proxy server to use for the first connection.

proxy_password

specifies the password needed to connect to a proxy server for the first connection.

proxy_port

specifies the port number where the proxy server is listening for requests for the first connection. The default is `0`.

proxy_user

specifies the user name needed to connect to a proxy server for the first connection.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for using a proxy server with OAuth 2.0 refresh token grant authentication.

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:googleanalytics:
    ProxyHost=pserver;ProxyPassword=secret;ProxyPort=808;ProxyUser=jsmith;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

Data types

The following table lists data types supported by the driver and how they are mapped to JDBC data types.

See "getTypeInfo()" for getTypeInfo() results of data types supported by the driver.

Table 1: Google Analytics Data Types

Google Analytics Data Type	JDBC Data Type
ARRAY	VARCHAR
BOOLEAN	BOOLEAN
CDSTYPE	VARCHAR
CURRENCY	DECIMAL
DATE	DATE
DATETIME	TIMESTAMP_WITH_TIMEZONE
DOUBLE	DOUBLE
FLOAT	DOUBLE
INTEGER	BIGINT
LONG	BIGINT
METADATATYPE	VARCHAR
PERCENT	DOUBLE
SAMPLINGLEVEL	VARCHAR
STRING	VARCHAR
TIME	DOUBLE
URL	VARCHAR

getTypeInfo()

The DatabaseMetaData.getTypeInfo() method returns information about data types. The following table provides getTypeInfo() results for supported data types.

Table 2: getTypeInfo() Results

TYPE_NAME = ARRAY AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = ARRAY MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 255 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = BOOLEAN AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 16 (BOOLEAN) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = BOOLEAN MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 1 SEARCHABLE = 3 SQL_DATA_TYPE = 16 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = CDSTYPE AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = CDSTYPE MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 18 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = CURRENCY AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 3 (DECIMAL) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = CURRENCY MAXIMUM_SCALE = 2	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 18 SEARCHABLE = 3 SQL_DATA_TYPE = 3 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = DATE AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 91 (DATE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = DATE ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = DATE MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 10 SEARCHABLE = 3 SQL_DATA_TYPE = 91 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = DATETIME AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 2014 (TIMESTAMP_WITH_TIMEZONE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = TIMESTAMP ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = DATETIME MAXIMUM_SCALE = 3	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 29 SEARCHABLE = 3 SQL_DATA_TYPE = 95 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = DOUBLE AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 8 (DOUBLE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = DOUBLE MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 15 SEARCHABLE = 3 SQL_DATA_TYPE = 8 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = FLOAT AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 8 (DOUBLE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = FLOAT MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 15 SEARCHABLE = 3 SQL_DATA_TYPE = 8 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = INTEGER AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -5 (BIGINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = INTEGER MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 19 SEARCHABLE = 3 SQL_DATA_TYPE = 25 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = LONG AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -5 (BIGINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = LONG MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 19 SEARCHABLE = 3 SQL_DATA_TYPE = 25 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = METADATATYPE AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = METADATATYPE MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 9 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = PERCENT AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 8 (DOUBLE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = PERCENT MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 15 SEARCHABLE = 3 SQL_DATA_TYPE = 8 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = SAMPLINGLEVEL AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = SAMPLINGLEVEL MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 16 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = STRING AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = STRING MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 255 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = TIME AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 8 (DOUBLE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = TIME MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 15 SEARCHABLE = 3 SQL_DATA_TYPE = 8 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = URL AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = URL MAXIMUM_SCALE = 0	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 255 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

SQL escape sequences

The driver supports the following SQL escape sequences.

- Date, Time, and Timestamp Escape Sequences
- Scalar Functions
- Outer Join Escape Sequences
- LIKE Escape Character Sequence for Wildcards

Refer to "SQL escape sequences" in the *Progress DataDirect for JDBC Drivers Reference* for information about SQL escape sequences.

Supported scalar functions

The driver supports the scalar functions in the following table. Note that your database system may not support all these functions. Refer to the documentation for your database system to find out which functions are supported by your database.

In addition, you can also determine the supported scalar functions by using DatabaseMetaData methods.

You can use scalar functions in SQL statements with the following syntax:

`{fn` *`scalar-function`*`}`

where:

`scalar-function`

is a scalar function supported by the drivers, as listed in the following table.

Example:

```
SELECT id, name FROM emp WHERE name LIKE {fn UCASE('Smith')}
```

Refer to "Scalar functions" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

Table 3: Supported Scalar Functions

String Functions	Numeric Functions	Timedate Functions	System Functions
ASCII	ABS	CURDATE	CURSESSIONID
BIT_LENGTH	ACOS	CURRENT_DATE	DATABASE
CHAR	ASIN	CURRENT_TIME	IDENTITY
CHAR_LENGTH	ATAN	CURRENT_TIMESTAMP	USER
CHARACTER_LENGTH	ATAN2	CURTIME	
CONCAT	BITAND	DATEDIFF	
DIFFERENCE	BITOR	DATE_ADD	
HEXTORAW	BITXOR	DATE_SUB	
INSERT	CEILING	DAY	
LCASE	COS	DAYNAME	
LEFT	COT	DAYOFMONTH	
LENGTH	DEGREES	DAYOFWEEK	
LOCATE	EXP	DAYOFYEAR	
LOCATE_2	FLOOR	EXTRACT	

String Functions	Numeric Functions	Timedate Functions	System Functions
LOWER	LOG	HOUR	
LTRIM	LOG10	MINUTE	
OCTET_LENGTH	MOD	MONTH	
RAWTOHEX	PI	MONTHNAME	
REPEAT	POWER	NOW	
REPLACE	RADIANS	QUARTER	
RIGHT	RAND	SECOND	
RTRIM	ROUND	SECONDS_SINCE_MIDNIGHT	
SOUNDEX	ROUNDMAGIC	TIMESTAMPADD	
SPACE	SIGN	TIMESTAMPDIFF	
SUBSTR	SIN	TO_CHAR	
SUBSTRING	SQRT	WEEK	
UCASE	TAN	YEAR	
UPPER	TRUNCATE		

DataDirect tools

Progress DataDirect for JDBC drivers install the set of tools described in this section. For detailed instructions on using these tools, refer to the corresponding topics in the *Progress DataDirect for JDBC Drivers Reference*.

- DataDirect Test allows you to test your JDBC driver and learn the JDBC API.
- DataDirect Connection Pool Manager allows you to pool connections when accessing databases. When your applications use connection pooling, connections are reused rather than created each time a connection is requested. Because establishing a connection is among the most costly operations an application may perform, using Connection Pool Manager to implement connection pooling can significantly improve performance.
- Statement Pool Monitor loads statements into and remove statements from the statement pool as well as generate information to help you troubleshoot statement pooling performance.
- DataDirect Spy logs detailed information about calls your driver makes that can be used for troubleshooting.

Troubleshooting

The *Progress DataDirect for JDBC Drivers Reference* provides information on troubleshooting problems should they occur. Refer to the "Troubleshooting" section in the *Reference* for details.

Contacting Technical Support

Progress DataDirect offers a variety of options to meet your support needs. Please visit our Web site for more details and for contact information:

https://www.progress.com/support

The Progress DataDirect Web site provides the latest support information through our global service network. The SupportLink program provides access to support contact details, tools, patches, and valuable information, including a list of FAQs for each product. In addition, you can search our Knowledgebase for technical bulletins and other information.

When you contact us for assistance, please provide the following information:

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full information, including location.
- The Progress DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your product.
- Any database, database version, third-party software, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be re-created.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

2

Tutorials

The following sections guide you through using the driver to access your data with some common third-party applications. For information on installing your driver and setting the CLASSPATH, see "Installing and setting-up the driver."

For details, see the following topics:

- Tableau
- DbVisualizer
- Interactive SQL for JDBC (JDBCISQL)

Tableau

After you have installed your driver and defined it on the CLASSPATH, you can use the driver to access your data with Tableau. Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data. By using the driver with Tableau, you can improve performance when retrieving data while leveraging the driver's relational mapping tools.

To use the driver to access data with Tableau:

1. Navigate to the `\lib\xx` subdirectory of the Progress DataDirect installation directory; then, locate the `jar` file for your driver:

   ```
   googleanalytics.jar
   ```

2. Copy the `.jar` file for your driver into the following directory:

Windows: `C:\Program Files\Tableau\Drivers`

Linux: `/opt/tableau/tableau_driver/jdbc`

3. Open Tableau. From the **Connect** menu, select **Other Databases (JDBC)**.
4. In the **Other Databases (JDBC)** dialog, provide values for the following fields; then, click **Sign In**.
   - **URL**: Copy and paste your connection URL into this field. The following example shows how to connect using the OAuth 2.0 refresh token grant.

     ```
     jdbc:datadirect:googleanalytics:ClientId=client_id;ClientSecret=client_secret;
     RefreshToken=refresh_token;
     ```

     Note: See OAuth 2.0 authentication on page 40 for details.

   - **Dialect**: Select **SQL92** (the default) from the drop-down box.
5. The **Data Source** window appears. In the **Schema** field, select the schema for the service you want to use.
6. In the **Table** field, the tables stored in the selected schema are now exposed and available for selection.

You have successfully accessed your data and are now ready to create reports with Tableau. For detailed information, refer to the Tableau product documentation at: https://www.tableau.com/support/help.

DbVisualizer

After you have installed your driver and defined it on the CLASSPATH, you can use the driver to access your data with the third-party DbVisualizer tool. The following topics guide you through using DbVisualizer to add your driver, connect, and execute SQL statements.

Adding a driver

To add a driver with DbVisualizer:

1. Open DbVisualizer.
2. From the menu, select **Tools**>**Driver Manager**. The Driver Manager window opens.
3. From the Driver Manager menu, select **Driver**>**Create Driver**.
4. Click the  button to navigate to the location of the driver jar file; then, click **OK**. The following are the default locations for the driver:

   Windows

   ```
   C:\Program Files\Progress\DataDirect\JDBC\lib\60\googleanalytics.jar
   ```

   Linux

   ```
   /opt/Progress/DataDirect/JDBC/lib/60/googleanalytics.jar
   ```

5. Provide values for the following fields; then, close the Driver Manager window.

- **Name**: Type an alias for your driver. For example:

  ```
  GoogleAnalytics
  ```

- **URL Format**: Optionally, specify the format of the connection string for your driver. For example:

  ```
  jdbc:datadirect:googleanalytics:AccessToken=access_token
  ```

- **Driver Class**: From the drop down menu, select the driver class for your driver. For example:

  ```
  com.ddtek.jdbc.googleanalytics.GoogleAnalyticsDriver
  ```

You can now use your driver with DbVisualizer. Proceed to "Connecting and executing SQL statements" for information on connecting and executing SQL statements.

Connecting and executing SQL statements

To use the driver to access data with DbVisualizer:

1. Open DbVisualizer.
2. From the menu, select **Database**>**New Connection**. When prompted to use the Connection Wizard, click **OK**.
3. Provide the following information when prompted; then, click **Next** to proceed:
   - **Connection alias**: Type the name to be used when referring to this connection.
   - **Driver**: Select the alias that you provided for your driver from the drop-down menu.
4. Copy and paste your connection URL into the **Database URL** field; then, click **Finish**.

   The following example shows how to connect using the OAuth 2.0 refresh token grant.

   ```
   jdbc:datadirect:googleanalytics:googleanalytics:ClientId=client_id;
   ClientSecret=client_secret;RefreshToken=refresh_token;
   ```

   Note: You can also generate connection strings using Google Analytics Configuration Manager. For more information, see Generating connection URLs with the Configuration Manager on page 33.

5. To execute SQL statements, select **SQL Commander**>**New SQL Commander**. A SQL Commander tab opens.
6. Select values for the following fields:
   - **Database Connection**: Select connection alias you provided for the connection from the drop-down menu.
   - **Schema**: Select the schema you want to execute queries against from the drop-down menu.
7. In the SQL Commander tab, enter SQL commands you want to execute; then select **SQL Commander**>**Execute**. For example:

   To select all of the rows from the ACCOUNT table:

   ```
   SELECT * FROM ACCOUNT
   ```

To select the URLs for a specified issue:

```
SELECT * FROM ACCOUNT WHERE ID = <ID>
```

See "Supported SQL statements and extensions" for the supported syntax used to execute SQL statements.

Note: If you are fetching large sets of data, you may want to limit the results using the Max Rows and Max Chars fields.

You have successfully accessed your data with DbVisualizer.

Interactive SQL for JDBC (JDBCISQL)

After you have installed your driver and defined it on the CLASSPATH, you can use the driver to access your data with Interactive SQL for JDBC (JDBCISQL). JDBCISQL supports a command line interface that allows you to connect to a data source, execute SQL statements and retrieve results for display on a terminal.

To execute commands with JDBCISQL:

1. Start the ISQL tool. Do one of the following:
   - On Windows, double-click the`jdbcisql.bat` file in the *`install_dir`*`\jdbcisql` folder. Or, from a command prompt, navigate to the *`install_dir`*`\jdbcisql` directory and run the `jdbcisql.bat` file.
   - On Linux and UNIX, change to the *`install_dir`*`\isql` directory and run `jdbcisql.sh`.

   The Interactive SQL prompt appears.
2. Type the driver name class; then, press **Enter**:

   ```
   com.ddtek.jdbc.googleanalytics.GoogleAnalyticsDriver
   ```

3. Type `connect` followed by the connection URL for the driver; then, press **Enter**. For example:

   ```
   connect jdbc:datadirect:googleanalytics:ClientId=client_id;
   ClientSecret=client_secret;RefreshToken=refresh_token;
   ```

   Note: If you are using the AddTables connection property with JDBCISQL, you must escape double quotes with a backslash when specifying the AddTables value. For example:

   ```
   AddTables='{\"sample_table\":[\"col1\",\"col2\",\"col3\"]}
   ```

   If successful, the tool will return the time required to connect.
4. At the `ISQL>` prompt, issue a SQL command to query or modify the data source; then, press **Enter**. For example:

   ```
   SELECT * FROM ACCOUNT;
   ```

   Note: SQL commands must be terminated by a semi-colon.

Note: In addition to SQL commands, JDBCISQL supports a set of proprietary commands. Type `Help` at the prompt for a list of supported commands and syntax.

The results of the command are displayed in the terminal.

5. After you are finished executing queries and commands, you can disconnect from the data source by typing the following; then, pressing **Enter**:

   ```
   DISCONNECT;
   ```

6. Press any key to end the session.

3

Configuring and connecting

This section provides information on how to connect to your data store using either the JDBC Driver Manager or DataDirect JDBC data sources, as well as information on how to implement and use functionality supported by the driver.

After the driver has been installed and defined on your classpath, you can connect from your application to your data in either of the following ways.

- Using the JDBC `DriverManager` by specifying the connection URL in the `DriverManager.getConnection()` method.
- Creating a JDBC data source that can be accessed through the Java Naming Directory Interface (JNDI).

For details, see the following topics:

Setting the classpath

The driver must be defined on your CLASSPATH before you can connect. The CLASSPATH is the search string your Java Virtual Machine (JVM) uses to locate JDBC drivers on your computer. If the driver is not defined on your CLASSPATH, you will receive a `class not found` exception when trying to load the driver. Set your system CLASSPATH to include the driver jar file as shown, where *`install_dir`* is the path to your product installation directory.

`install_dir``/lib/60/googleanalytics.jar`

Windows Example

```
CLASSPATH=.;C:\Program Files\Progress\DataDirect\JDBC\lib\60\googleanalytics.jar
```

UNIX Example

```
CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/googleanalytics.jar
```

Connecting using the JDBC Driver Manager

One way to connect to a service is through the JDBC DriverManager using the `DriverManager.getConnection()` method. As the following examples show, this method specifies a string containing a connection URL.

OAuth 2.0 access token flow

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:googleanalytics:AuthenticationMethod=OAuth2;
    ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;");
```

OAuth 2.0 refresh token grant

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:googleanalytics:AuthenticationMethod=OAuth2;
    ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

Note: See OAuth 2.0 authentication on page 40 for details.

Passing the connection URL

After setting the CLASSPATH, the required connection information needs to be passed in the form of a connection URL. The following example includes the properties required for connecting with OAuth 2.0 refresh token grant authentication.

Connection URL Syntax

The connection URL takes the following form:

```
jdbc:datadirect:googleanalytics:AuthenticationMethod=OAuth2;
ClientID=client_id;ClientSecret=client_secret;
RefreshToken=refresh_token;[property=value[;...]];
```

where:

client_id

specifies the client ID key for your application when authenticating with OAuth 2.0.

client_secret

specifies the client secret for your application when authenticating with OAuth 2.0.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for connecting with the OAuth 2.0 authentication.

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googleanalytics:AuthenticationMethod=OAuth2;
  ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
  ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
  RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

See also

Connection property descriptions on page 47
Connection URL examples on page 12

Generating connection URLs with the Configuration Manager

The driver includes a browser-based tool, Progress DataDirect Google Analytics Configuration Manager, that allows you to generate connection URLs, test connections, and execute test queries. This section will guide you through generating and testing a connection URL that can be used by your application.

To generate a connection URL:

1. Open the Google Analytics Configuration Manager by double-clicking the driver jar file. Or, in a command line, navigate to the directory containing your driver jar file; then, execute the following command:

   ```
   java -jar googleanalytics.jar
   ```

The Google Analytics Configuration Manager opens in your default web browser.

2. From the browser window, provide values of the connection properties you want to configure in the corresponding fields. A connection URL will generate in the Connection String field as you provide settings. To view more properties, select the tabs at the top of the page. See "Connection URL examples" for a list of required properties and properties used for different configurations.

   Note: If you do not specify a value for an optional property, the property will be omitted from the string and the default value will be used.

3. Optionally, add a new table to the relational schema of your Google Analytics data, or customize a predefined custom table.

   For more information about custom tables, and how they can be used to write useful queries, see Introduction to the Google Analytics Data Model.

   a) Under the **Schema Settings** tab, click **Configure Logical Schema**.

   b) Select a custom table from the dropdown list, or click **Create Table** and type the name of a new table.

   c) Select the metrics and dimensions you want to expose with the table. You may specify up to ten metrics and seven dimensions per table.

   d) Click **Save & Close**.

   e) The **Add Tables** field is populated with the table definition in the form of a JSON string.

4. Optionally, you can manually edit your string by clicking the Edit button ().
5. At any point during the process, you can click **Test Connect** to attempt to connect to the service using the string generated in the Connection String field. In the **Test Connection** window:

   a) Provide values for any fields required by your service.

   b) Optionally, in the Test Query field, enter any SQL queries you want to execute during the test. For example:

   ```
   SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
   ```

   c) Click **Execute**.

   If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

   Note: The information you enter in the logon dialog box during a test connect is not saved.

6. To use your string, click the Copy button () and paste the string to a location that can be used by your application.

See also

OAuth 2.0 authentication on page 40
Connection URL examples on page 12

Testing connections and queries

You can quickly test a connection string and queries using Progress DataDirect Google Analytics Configuration Manager.

To test your connection and query:

1. Open the Google Analytics Configuration Manager by double-clicking on the driver jar file. Or, in a command line, navigate to the directory containing your driver jar file; then, execute the following command:

   ```
   java -jar googleanalytics.jar
   ```

   The Google Analytics Configuration Manager opens in your default web browser.

2. Provide a connection string to test using one of the following methods:

   - Entering a string: Click the Edit button (); then, paste your string into the Connection String field. If you prefer, you can also type a string directly into this field.
   - Generating a string: Provide values of the connection properties you want to configure into the corresponding fields. The Google Analytics Configuration Manager will generate a string in the Connection String field based on the values you specify.

3. Click **Test Connect** to attempt to connect to the service using the string specified in the Connection String field. The **Test Connection** window appears.
4. Provide connection property values for any fields required by your service.
5. To execute a test query with the test connection, enter a SQL query into the Test Query field. For example:

   ```
   SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
   ```

6. Click **Execute**.

   If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

Connecting using data sources

A *JDBC data source* is a Java object, specifically a `DataSource` object, that defines connection information required for a JDBC driver to connect to the database. Each JDBC driver vendor provides their own data source implementation for this purpose. A Progress DataDirect data source is Progress DataDirect's implementation of a `DataSource` object that provides the connection information needed for the driver to connect to a database.

Because data sources work with the Java Naming Directory Interface (JNDI) naming service, data sources can be created and managed separately from the applications that use them. Because the connection information is defined outside of the application, the effort to reconfigure your infrastructure when a change is made is minimized. For example, if the database is moved to another database server, the administrator need only change the relevant properties of the `DataSource` object. The applications using the database do not need to change because they only refer to the name of the data source.

How data sources are implemented

Data sources are implemented through a `DataSource` class. A data source class implements the following interfaces.

- `javax.sql.DataSource`
- `javax.sql.ConnectionPoolDataSource` (allows applications to use connection pooling)

Refer to "Connection Pool Manager" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

See also

Driver and DataSource classes on page 12

Creating data sources

The following example files provide details on creating and using Progress DataDirect data sources with the Java Naming Directory Interface (JNDI), where *`install_dir`* is the product installation directory.

- *`install_dir`*`/Examples/JNDI/JNDI_LDAP_Example.java` can be used to create a JDBC data source and save it in your LDAP directory using the JNDI Provider for LDAP.
- *`install_dir`*`/Examples/JNDI/JNDI_FILESYSTEM_Example.java` can be used to create a JDBC data source and save it in your local file system using the File System JNDI Provider.

See "Example data source" for an example data source definition for the example files.

To connect using a JNDI data source, the driver needs to access a JNDI data store to persist the data source information. For a JNDI file system implementation, you must download the File System Service Provider from the Oracle Technology Network Java SE Support downloads page, unzip the files to an appropriate location, and add the `fscontext.jar` and `providerutil.jar` files to your CLASSPATH. These steps are not required for LDAP implementations because the LDAP Service Provider is included with supported versions of Java SE.

Example data source

To configure a data source using the example files, you will need to create a data source definition. The content required to create a data source definition is divided into three sections.

First, you will need to import the data source class. For example:

```
import com.ddtek.jdbcx.googleanalytics.GoogleAnalyticsDataSource;
```

Next, you will need to set the values and define the data source. For example, the following definition contains the minimum properties required for a connection using the OAuth 2.0 refresh token grant.

Note: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

Note: In a JDBC data source, string values must be enclosed in double quotation marks, for example, `setProxyPassword("secret")`.

```
GoogleAnalyticsDataSource mds = new GoogleAnalyticsDataSource();
mds.setDescription("My Google Analytics Data Source");
mds.AuthenticationMethod("OAuth2");
mds.ClientID("29453d6f-6789-25gh-gd8g-44tk3c527831");
mds.ClientSecret("12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd");
mds.RefreshToken("12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831");
```

Finally, you will need to configure the example application to print out the data source attributes. Note that this code is specific to the driver and should only be used in the example application. For example, you would add the following section for the minimum properties required to establish a connection:

```
if (ds instanceof GoogleAnalyticsDataSource)
{
GoogleAnalyticsDataSource jmds = (GoogleAnalyticsDataSource) ds;
System.out.println("description=" + jmds.getDescription());
System.out.println("authenticationmethod=" + jmds.getAuthenticationMethod());
System.out.println("clientid=" + jmds.getClientID());
...
System.out.println();
}
```

Calling a data source in an application

Applications can call a Progress DataDirect data source using a logical name to retrieve the `javax.sql.DataSource` object. This object loads the specified driver and can be used to establish a connection to the database.

Once the data source has been registered with JNDI, it can be used by your JDBC application as shown in the following code example.

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("EmployeeDB");
Connection con = ds.getConnection("domino", "spark");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the data source (`EmployeeDB`). The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.DataSource` object. Then, the `DataSource.getConnection()` method is called to establish a connection.

Testing a data source connection

You can use DataDirect Test™ to establish and test a data source connection. The screen shots in this section were taken on a Windows system.

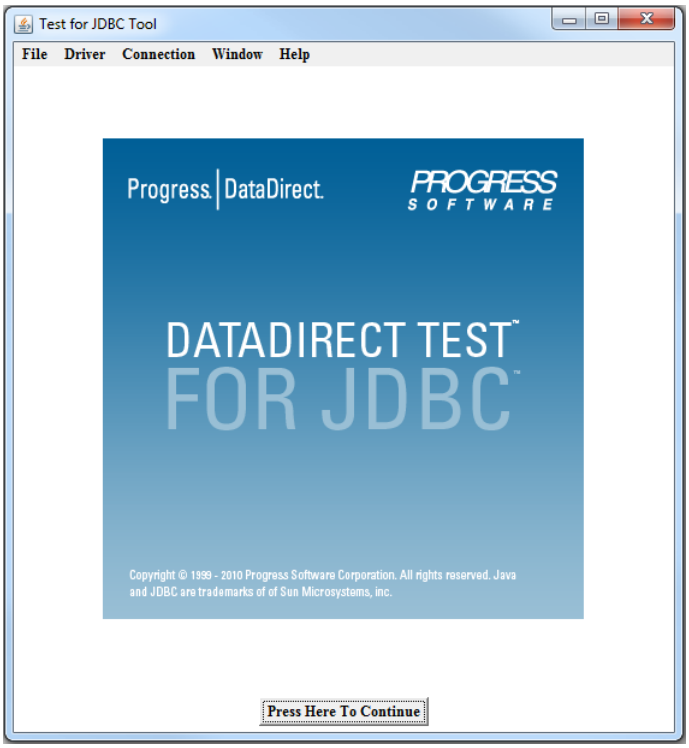
Take the following steps to establish a connection.

1. Navigate to the installation directory. The default location is:
   - Windows systems: `Program Files\Progress\DataDirect\JDBC\testforjdbc`
   - UNIX and Linux systems: `/opt/Progress/DataDirect/JDBC/testforjdbc`

Note: For UNIX/Linux, if you do not have access to `/opt`, your home directory will be used in its place.

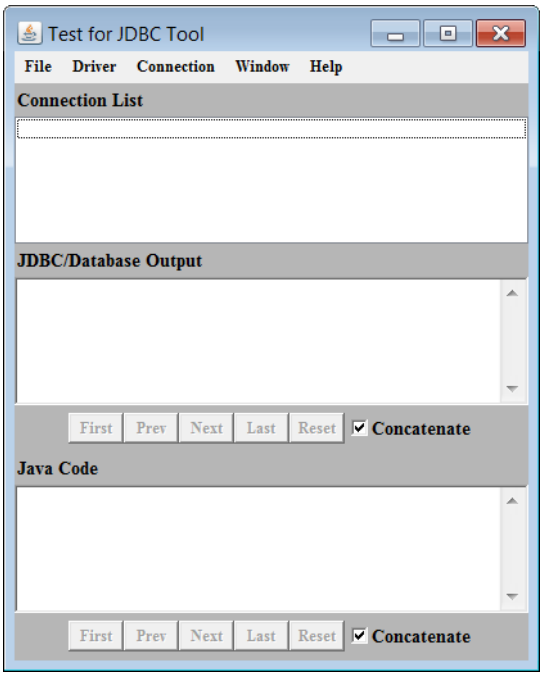
2. From the `testforjdbc` folder, run the platform-specific tool:
   - `testforjdbc.bat` (on Windows systems)
   - `testforjdbc.sh` (on UNIX and Linux systems)

   The **Test for JDBC Tool** window appears:



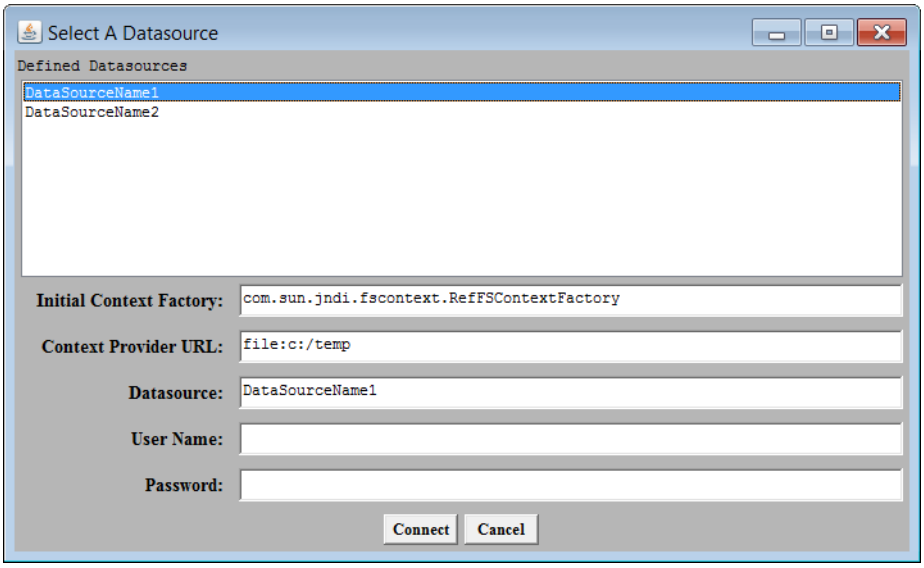
3. Click **Press Here to Continue**.

   The main dialog appears:



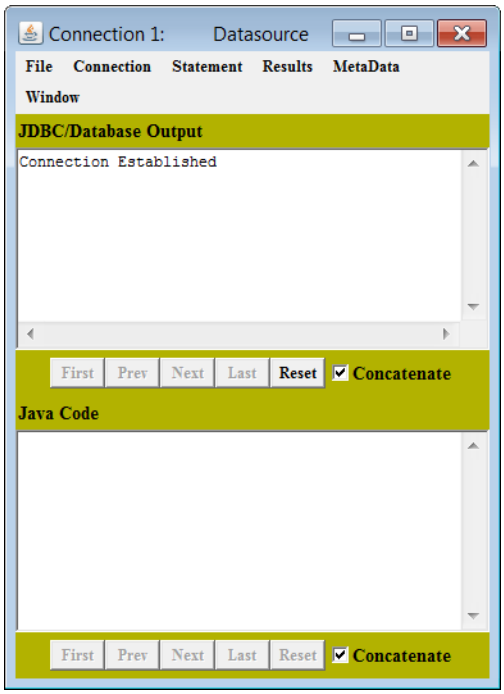
4. From the menu bar, select **Connection > Connect to DB via Data Source**.

   The **Select A Database** dialog appears:



5. Select a datasource template from the **Defined Datasources** field.
6. Provide the following information:
   a) In the **Initial Context Factory**, specify the location of the initial context provider for your application.
   b) In the **Context Provider URL**, specify the location of the context provider for your application.
   c) In the **Datasource** field, specify the name of your datasource.
7. If you are using user ID/password authentication, enter your user ID and password in the corresponding fields.
8. Click **Connect**.

If the connection information is entered correctly, the **JDBC/Database Output** window reports that a connection has been established. If a connection is not established, the window reports an error.



OAuth 2.0 authentication

The driver supports OAuth 2.0 to access Google Analytics resources. Before you can configure the driver for OAuth, you must register your client application with Google Cloud and obtain information such as the client ID, the client secret, and the refresh token. The following workflow describes the process for setting up OAuth 2.0 access.

1. Register your application with Google Cloud. See this topic for step-by-step instructions for registering your application.
2. Obtain client ID and client secret. If the application has already been registered, see this topic for steps to obtain the client ID and client secret.
3. Obtain access and refresh tokens using the Configuration Manager. To configure the driver, you will need to specify either an access token or a refresh token. This topic provides instructions to obtain these tokens using the driver's Configuration Manager.
4. Configure the driver to use OAuth 2.0. You can configure the driver to access Google Analytics resources by specifying the connection properties described in this topic.

Note: For more information, refer to the Using OAuth 2.0 to Access Google APIs section of the *Google Identity Help*.

Registering your application with Google Cloud

Registering your client application with Google Cloud involves three basic steps: creating a Google Project, registering the application, and creating OAuth credentials.

- Create a Google Cloud project
- Register the application
- Create OAuth credentials

Create a Google Cloud project

If you do not already have a Google Cloud project to access Google Analytics resources, you must create one to register a client application.

Take the following steps to create a Google Cloud project.

1. Go to Google Developer Console.
2. Navigate to **Enabled APIs & services**.
3. Create the project.
   - **Option 1.** If you have no projects, click **CREATE PROJECT**.
   - **Option 2.** If you are in a project but want to create a new one, select the project you are in, and then click **NEW PROJECT**.
4. Specify the project name in the **Project name** field.
5. Specify the location in the **Location** field.
6. Click **CREATE**.

Results:

You have created a Google Cloud project. From within this project, you will register your client application and create OAuth credentials.

Register the application

Take the following steps to register an application to access the resources of a Google Cloud project.

1. From the Google Developer Console, navigate to the **APIs & Services > OAuth Consent screen**.
2. Select the **User Type**.
3. Click **CREATE**.
4. Complete the application registration forms.

   OAuth Consent Screen form

   a. Specify values for the following fields.
   - App name
   - User support email
   - App logo
   - Application home page
   - Application privacy policy link

- Application terms of service link
- Authorized domain
- Developer contact information

b. Click **SAVE AND CONTINUE**.

Scopes form

a. Click **ADD OR REMOVE SCOPES**.

b. Select the scope or scopes to specify permissions for the client application. In some cases, you may need to manually enter scopes. The following scopes are the default scopes used by the driver and provide read-only access to Google Analytics resources.

- **View and download data:**

  `https://www.googleapis.com/auth/analytics.readonly`

- **View user permissions:**

  `https://www.googleapis.com/auth/analytics.manage.users.readonly`

Note: The values you select should be saved to a secure location. You will need to specify these values to obtain access and refresh tokens, as described in Obtain access and refresh tokens using the Configuration Manager.

---

c. Click **UPDATE**.

d. Click **SAVE AND CONTINUE**.

Test users form

a. Click **ADD USERS**.

b. Enter email address of each test user.

c. Click **ADD**.

d. Click **SAVE AND CONTINUE**.

Summary Page

a. Review the summary.

b. Click **BACK TO DASHBOARD**.

Results:

The OAuth consent screen page for your application is displayed. You have completed the registration process for your application.

Create OAuth credentials

Take the following steps to create client ID and client secret OAuth credentials for your application.

1. From the Google Developer Console, navigate to the **Credentials** screen by clicking **Credentials** on the left.
2. Click **CREATE CREDENTIALS** and select **OAuth client ID**.
3. Select an application type, and enter the requested information.

Note: The **Desktop app** option is a common option for initially setting up and testing the driver from `localhost`. If you are using Progress DataDirect Hybrid Data Pipeline, choose **Web application** even if you have deployed Hybrid Data Pipeline on a local machine.

4. Click **CREATE**.

   Step result: A dialog is displayed with the client ID and secret.

5. Save the application information to a secure location.

   - **Option 1.** Download the JSON file and save it to a secure location. The JSON file includes the client ID and secret as well as summary information about the client application, including useful endpoints.
   - **Option 2.** Copy the client ID and secret, and save them to a secure location.

Note: The values for the client ID and secret should be saved to a secure location. You will need to specify these values for the ClientID and ClientSecret connection properties.

Results:

You have created OAuth credentials for your client application.

Obtaining the client ID and client secret

Take the following steps to obtain the client ID and secret for an application that has already been registered with Google Cloud.

1. Go to the Google Developer Console.
2. Select the Google Cloud project in which the client application has been registered.
3. Select **Credentials** on the left.
4. Under **OAuth 2.0 Client IDs**, select the client application for which you are retrieving the client ID and secret.
5. The client ID and secret appear on the upper right of the screen. Copy this information to a secure location.

Results:

You have obtained OAuth credentials for your client application.

Obtain access and refresh tokens using the Configuration Manager

You need the following information before you begin.

- The client ID and client secret for the client application
- The scope or scopes that define permissions for the client application

The Configuration Manager uses the authorization code grant to obtain access and refresh tokens from Google Cloud. The following steps describe how you can use the Configuration Manager to obtain access and refresh tokens. In addition, the Configuration Manager produces a connection URL that you can use in your application.

Note: You must allow popups in your browser to obtain access and refresh tokens with the Configuration Manager.

1. Open the Configuration Manager by double-clicking the driver jar file. Alternately, from the command line, navigate to the directory containing your driver jar file, and then execute the following command:

   ```
   java -jar googleanalytics.jar
   ```

   Step result: The Configuration Manager opens in your default web browser.

2. Enter values for the required connection properties under the **Connection** tab.
   - **Client ID**
   - **Client Secret**
   - **Scope**

     The following scopes are the default scopes used by the driver.
     - **View and download data:**

       `https://www.googleapis.com/auth/analytics.readonly`
     - **View user permissions:**

       `https://www.googleapis.com/auth/analytics.manage.users.readonly`

3. Enter any additional values under the **Connection** tab, or other tabs, as desired.
4. Retrieve access and refresh tokens.
   a. Click **Fetch OAuth Token**.
   b. If prompted, enter Google credentials.
   c. Provide consent to allow the Configuration Manager to retrieve the tokens.
   d. The **Access Token** and **Refresh Token** fields populate with values retrieved from Google Cloud.
5. Click **Test Connect** to verify connectivity and run SQL queries against the service.

Results:

The **Access Token** and **Refresh Token** fields contain access and refresh tokens. You can use these tokens to configure the driver and access Google Analytics resources.

The connection string in the **Connection String** field may be copied and used in your JDBC application to access Google Analytics resources.

Note: Not all the values in the resulting connection string may be required. However, the connection string can be copied directly into your JDBC application. The driver ignores any values that are not required.

Configuring the driver to use OAuth 2.0

Prerequisites:

- Client application registered with Google Cloud
- An access token or refresh token

After you have registered your client application with Google Cloud and obtained the required OAuth information, you may configure the driver to access Google Analytics resources using OAuth 2.0.

To configure the driver, you must provide either the AccessToken or the RefreshToken in the connection URL.

- Set the AccessToken property to specify the access token you have obtained from Google. Generally, an access token is available for a short period of time and may only be used for a single session.
- Set the RefreshToken property to specify the refresh token you have obtained from Google. With a valid refresh token, the driver can obtain a new access token. In this way, a refresh token enables application access to Google for multiple sessions over an extended period of time.

The AccessToken and RefreshToken properties may both be specified. If a value for the AccessToken property is not specified, the driver uses the value of the RefreshToken property to make a connection. If both values are not specified, the driver cannot make a successful connection. If both are specified, the driver ignores the AccessToken value and uses the RefreshToken value to generate a new AccessToken value.

See Obtain access and refresh tokens using the Configuration Manager for details on how to obtain an access or refresh token using the driver's Configuration Manager.

Access token example URL

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googleanalytics:AccessToken=access-token;");
```

Refresh token example URL

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googleanalytics:RefreshToken=refresh-token;");
```

Performance considerations

FetchSize: FetchSize can be used to adjust the trade-off between throughput and response time. In general, setting larger values for FetchSize will improve throughput, but can reduce response time. You should set FetchSize to suit your environment. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes improve overall fetch times at the cost of additional memory.

InsensitiveResultSetBufferSize: To improve performance, result set data can be cached instead of written to disk. If the size of the result set data is greater than the size allocated for the cache, the driver writes the result set to disk. The maximum cache size setting is 2 GB.

MaxPooledStatements: To improve performance, the driver's own internal prepared statement pooling should be enabled when the driver does not run from within an application server or from within another application that does not provide its own prepared statement pooling. When the driver's internal prepared statement pooling is enabled, the driver caches a certain number of prepared statements created by an application. For example, if the MaxPooledStatements property is set to `20`, the driver caches the last 20 prepared statements created by the application. If the value set for this property is greater than the number of prepared statements used by the application, all prepared statements are cached.

See also

FetchSize on page 61
InsensitiveResultSetBufferSize on page 63
MaxPooledStatements on page 65

4

Connection property descriptions

You can use connection properties to customize the driver for your environment. This section organizes connection properties according to functionality. You can use connection properties with either the JDBC `DriverManager` or a JDBC data source. For a `DriverManager` connection, a property is expressed as a key value pair and takes the form *`property=value`*. For a data source connection, a property is expressed as a JDBC method and takes the form `set`*`property`*`(`*`value`*`)`.

Note:

- In a JDBC data source, string values must be enclosed in double quotation marks, for example, `setProxyPassword("secret")`.
- The data type listed for each connection property is the Java data type used for the property value in a JDBC data source.
- Connection property names are case-insensitive. For example, `AuthenticationMethod` is the same as `authenticationmethod`.
- For connection properties that support string values, use the following escape sequence to specify values containing leading or trailing spaces and curly brackets: `{`*`value`*`}`. For example: `ProxyPassword={hello }` or `ProxyPassword={{hello}}`.

The following tables describe the connection properties by functionality.

- Mapping properties
- Statement pooling properties
- Additional properties

OAuth 2.0 properties

The following table summarizes properties used for OAuth 2.0 authentication method.

Property	Data Source Method	Default
AccessToken on page 52	`getAccessToken()` `setAccessToken(String)`	No default value
AuthenticationMethod on page 54	`getAuthenticationMethod()` `setAuthenticationMethod(String)`	`OAuth2`
ClientID on page 54	`getClientId()` `setClientId(String)`	No default value
ClientSecret on page 55	`getClientSecret()` `setClientSecret(String)`	No default value
RefreshToken on page 70	`getRefreshToken()` `setRefreshToken(String)`	No default value
Scope on page 73	`getOauthScope()` `setOauthScope(String)`	For default, see the connection property description.

Proxy server properties

The following table summarizes proxy server connection properties.

Property	Data Source Method	Default
ProxyHost on page 67	`getProxyHost()` `setProxyHost(String)`	No default value
ProxyPassword on page 68	`getProxyPassword()` `setProxyPassword(String)`	No default value

Property	Data Source Method	Default
ProxyPort on page 66	`getProxyPort()` `setProxyPort(Integer)`	`0` which means the default is determined by the ProxyHost property. For HTTP URLs: `80` For HTTPS URLs: `443`
ProxyUser on page 68	`getProxyUser()` `setProxyUser(String)`	No default value

Timeout properties

The following table summarizes timeout connection properties.

Property	Data Source Method	Default
LoginTimeout on page 65	`getLoginTimeout()` `setLoginTimeout(Integer)`	`0` (seconds)
QueryTimeout on page 69	`getQueryTimeout()` `setQueryTimeout(Integer)`	`0` (seconds)

Web service properties

The following table summarizes Web service connection properties, including those related to timeouts.

Property	Data Source Method	Default
LoginTimeout on page 65	`getLoginTimeout()` `setLoginTimeout(Integer)`	`0` (no timeout)
QueryTimeout on page 69	`getQueryTimeout()` `setQueryTimeout(Integer)`	`0` (no timeout)
StmtCallLimit on page 75	`getStmtCallLimit()` `setStmtCallLimit(Integer)`	`0` (no limit)
StmtCallLimitBehavior on page 75	`getStmtCallLimitBehavior()` `setStmtCallLimitBehavior(String)`	`ErrorAlways`

Mapping properties

The following table summarizes connection properties involved in mapping the Google Analytics data model to a SQL model.

Property	Data Source Method	Default
AddTables on page 53	`getAddTables()` `setAddTables(String)`	No default value
CustomPrefix on page 58	`getCustomPrefix()` `setCustomPrefix(String)`	`Include`
CustomPrefixName on page 58	`getCustomPrefixName()` `setCustomPrefixName(String)`	No default value
KeywordConflictSuffix on page 64	`getKeywordConflictSuffix()` `setKeywordConflictSuffix(String)`	No default value
SchemaMap on page 71	`getSchemaMap()` `setSchemaMap(String)`	Default value depends on environment
ShowDeprecatedObjects on page 73	`getShowDeprecatedObjects()` `setShowDeprecatedObjects(Boolean)`	`false`
ShowInternalTables on page 74	`getShowInternalTables()` `setShowInternalTables(Boolean)`	`false`
SubtractTables on page 76	`getSubtractTables()` `setSubtractTables(String)`	No default value

Statement pooling properties

The following table summarizes statement pooling connection properties.

Property	Data Source Method	Default
ImportStatementPool on page 62	`getImportStatementPool()` `setImportStatementPool(String)`	No default value
MaxPooledStatements on page 65	`getMaxPooledStatements()` `setMaxPooledStatements(Integer)`	`0`
RegisterStatementPoolMonitorMBean on page 70	`getRegisterStatementPoolMonitorMbean()` `setRegisterStatementPoolMonitorMbean(Boolean)`	`false`

Additional properties

The following table summarizes additional connection properties.

Property	Data Source Method	Default
AddTables on page 53	`getAddTables()` `setAddTables(String)`	No default value
ConnectionRetryCount on page 56	`getConnectionRetryCount()` `setConnectionRetryCount(Integer)`	5
ConnectionRetryDelay on page 57	`getConnectionRetryDelay()` `setConnectionRetryDelay(Integer)`	1 (second)

DefaultQueryOptions on page 59	`getDefaultQueryOptions()`
	`setDefaultQueryOptions(String)`
		If no value is specified (the default), the driver uses
		`startDate=30daysAgo;`
		`endDate=yesterday`

DefaultView on page 60	`getDefaultView()` `setDefaultView(String)`	No default value
FetchSize on page 61	`getFetchSize()` `setFetchSize(Integer)`	`100` (rows)
InitializationString on page 62	`getInitializationString()` `setInitializationString(String)`	No default value
InsensitiveResultSetBufferSize on page 63	`getInsensitiveResultsetBufferSize()` `setInsensitiveResultsetBufferSize(Integer)`	`2048` (KB of memory)
TransactionMode on page 77	`getTransactionMode()` `setTransactionMode(String)`	`NoTransactions`

For details, see the following topics:

- AccessToken
- AddTables
- AuthenticationMethod
- ClientID
- ClientSecret
- ConnectionRetryCount
- ConnectionRetryDelay
- CustomPrefix

AccessToken

Purpose

Specifies the access token used to authenticate to Google Analytics with OAuth 2.0 enabled. Typically, this property is configured by the application; however, in some scenarios, you may need to secure a token using external processes. In those instances, you can also use this property to set the access token manually.

Valid Values

String

where:

String

is an access token you have obtained from the authentication service.

Notes

- Access tokens expire ten minutes after generation. Once connected, the access token remains valid till the session is disconnected.
- See "OAuth 2.0 authentication" for examples and more information.

Data Source Methods

```
public String getAccessToken()
```

```
public void setAccessToken(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 authentication on page 40

AddTables

Purpose

Specifies a JSON string that defines the tables, including dimensions and metrics, that appear in the Google Analytics schema. The JSON string must be defined using the Configuration Manager under the Schema Settings tab. Up to 7 dimensions and 10 metrics can be added to each table.

Valid Values

String

where:

String

is a JSON string that defines the tables that appear in the Google Analytics schema.

Example

```
{"TestTable":["adxImpressions","adxCoverage","_adGroup"]}
```

Data Source Methods

```
public String getAddTables()
public void setAddTables(String)
```

Default Value

No default value

Data Type

String

See also

Introduction to the Google Analytics Data Model on page 103

AuthenticationMethod

Purpose

Determines which authentication method the driver uses during the course of a session. OAuth 2.0 is currently the only supported authentication method. Therefore, AuthenticationMethod does not need to be specified. However, it does appear on the Configuration Manager's **Connection** tab.

Valid Values

`OAuth2`

Data Source Methods

```
public String getAuthenticationMethod()
public void setAuthenticationMethod(String)
```

Default Value

`OAuth2`

Data Type

String

See also

OAuth 2.0 authentication on page 40

ClientID

Purpose

Specifies the client ID key for your application when authenticating to Google Analytics with OAuth 2.0 enabled.

Valid Values

String

where:

String

is the client ID key for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getClientId()
```

```
public void setClientId(String)
```

Default Value

No default value

Data Type

String

See also

ClientSecret on page 55
OAuth 2.0 authentication on page 40

ClientSecret

Purpose

Specifies the client secret for your application when authenticating to Google Analytics with OAuth 2.0 enabled.

Important: The client secret is a confidential value used to authenticate the application to the service. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the client secret for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getClientSecret()
```

```
public void setClientSecret(String)
```

Default Value

No default value

Data Type

String

See also

ClientID on page 54
OAuth 2.0 authentication on page 40

ConnectionRetryCount

Purpose

The number of times the driver retries connection attempts to Google Analytics until a successful connection is established.

Valid Values

`0` | *x*

where:

x

is a positive integer that represents the number of retries.

Behavior

If set to `0`, the driver does not try to reconnect after the initial unsuccessful attempt.

If set to *x*, the driver retries connection attempts the specified number of times. If a connection is not established during the retry attempts, the driver returns an exception that is generated by the last server to which it tried to connect.

Example

If this property is set to `2`, the driver retries the server twice after the initial retry attempt.

Notes

- If an application sets a login timeout value (for example, using DataSource.loginTimeout or DriverManager.loginTimeout), and the login timeout expires, the driver ceases connection attempts.
- The ConnectionRetryDelay property specifies the wait interval, in seconds, to occur between retry attempts.

Data Source Methods

```
public Integer getConnectionRetryCount()
public void setConnectionRetryCount(Integer)
```

Default Value

5

Data Type

Integer

See also

ConnectionRetryDelay on page 57

ConnectionRetryDelay

Purpose

The number of seconds the driver waits between connection retry attempts when ConnectionRetryCount is set to a positive integer.

Valid Values

`0` | *x*

where:

x

is a number of seconds.

Behavior

If set to `0`, the driver does not delay between retries.

If set to *x*, the driver waits between connection retry attempts the specified number of seconds.

Example

If ConnectionRetryCount is set to 2 and this property is set to 3, the driver retries the server twice after the initial retry attempt. The driver waits 3 seconds between retry attempts.

Data Source Methods

```
public Integer getConnectionRetryDelay()
```

```
public void setConnectionRetryDelay(Integer)
```

Default Value

1

Data Type

Integer

See also

ConnectionRetryCount on page 56

CustomPrefix

Purpose

Determines whether the driver includes or strips a prefix denoting user-defined objects from table and column names when mapping the Google Analytics data model.

Valid Values

`Strip` | `Include`

Behavior

If set to `Strip`, the driver strips the prefix.

If set to `Include`, the driver includes the prefix.

Notes

When this property is set to `Strip`, the driver removes the prefix `new_` from any field or object that begins with this string. This property may be used to specify a prefix other than `new_`.

Data Source Methods

```
public String getCustomPrefix()
```

```
public void setCustomPrefix(String)
```

Default Value

`Include`

Data Type

String

CustomPrefixName

Purpose

Specifies the prefix that the driver removes from table and column names. When CustomPrefix is set to `Strip`, the driver removes this prefix from objects and fields when mapping the Google Analytics data model.

Valid Values

`String`

where:

`String`

is a string of alphanumeric characters.

Data Source Methods

`public String getCustomPrefixName()`

`public void setCustomPrefixName(String)`

Default Value

No default value

Data Type

String

DefaultQueryOptions

Purpose

Specifies the values of Google Analytics `startDate`, `endDate`, and `viewId` query parameters for WHERE clause filtering during a session. Providing values for these parameters simplifies queries.

Valid Values

`(key=value[;key=value])`

where:

`key`

is one of the following query parameters:

- `startDate`: the inclusive starting date for the query. The default is `30daysago` (thirty days prior to the current date).
- `endDate`: the inclusive ending date for the query. The default is `yesterday` (the day prior to the current date).
- `viewId`: a comma-separated list of Google Analytics View IDs. A View ID can be obtained from your Google Analytics dashboard (Dashboard>Admin>View Settings>View ID). There is no default..

  Important: In order for `SELECT * FROM` to work , `viewId` must be specified. If `viewId` is not specified using DefaultQueryOptions, then it must be set explicitly in the WHERE clause.

Notes

- The syntax for `startDate` and `endDate` values is as follows:
  - A date in the format *`YYYY-MM-DD`*
  - The word `today` for the current date
  - The word `yesterday` for the day prior to the current date
  - *nn*`daysAgo` where *nn* is a number of days prior to the current date

Data Source Methods

`public String getDefaultQueryOptions()`

```
public void setDefaultQueryOptions(String)
```

Default Value

If no value is specified (the default), the driver uses `startDate=30daysAgo;endDate=yesterday`.

Data Type

String

See also

Introduction to the Google Analytics Data Model on page 103

DefaultView

Purpose

Specifies the default view when connecting to and retrieving data from Google Analytics.

Valid Values

`String`

where:

`String`

is the name of a Google Analytics view.

Notes

The value of this string should be the same as the value in the View Name field (Google Analytics Dashboard>Admin>View Settings>View Name). The names of views can also be found via the Google Analytics VIEW table.

Data Source Methods

```
public String getDefaultView()
```

```
public void setDefaultView(String)
```

Default Value

No default value

Data Type

String

See also

Introduction to the Google Analytics Data Model on page 103

FetchSize

Purpose

Specifies the maximum number of rows that the driver processes before returning data to the application when executing a Select. This value provides a suggestion to the driver as to the number of rows it should internally process before returning control to the application. The driver may fetch fewer rows to conserve memory when processing exceptionally wide rows.

Valid Values

`0` | *x*

where:

x

is a positive integer indicating the number of rows that should be processed.

Behavior

If set to `0`, the driver processes all the rows of the result before returning control to the application. When large data sets are being processed, setting FetchSize to `0` can diminish performance and increase the likelihood of out-of-memory errors.

If set to *x*, the driver limits the number of rows that may be processed for each fetch request before returning control to the application.

Notes

- To optimize throughput and conserve memory, the driver uses an internal algorithm to determine how many rows should be processed based on the width of rows in the result set. Therefore, the driver may process fewer rows than specified by FetchSize when the result set contains exceptionally wide rows. Alternatively, the driver processes the number of rows specified by FetchSize when the result set contains rows of unexceptional width.
- FetchSize can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.
- You can use FetchSize to reduce demands on memory and decrease the likelihood of out-of-memory errors. Simply, decrease FetchSize to reduce the number of rows the driver is required to process before returning data to the application.

Data Source Methods

```
public Integer getFetchSize()
```

```
public void setFetchSize(Integer)
```

Default Value

`100`

Data Type

Integer

See also

Performance considerations on page 45

ImportStatementPool

Purpose

Specifies the path and file name of the file to be used to load the contents of the statement pool. When this property is specified, statements are imported into the statement pool from the specified file.

Valid Values

String

where:

String

is the path and file name of the file to be used to load the contents of the statement pool.

Notes

- If the driver cannot locate the specified file when establishing the connection, the connection fails and the driver throws an exception.
- For more information, refer to "Statement Pool Monitor" in the *Progress DataDirect for JDBC Drivers Reference*.

Data Source Methods

```
public String getImportStatementPool()
```

```
public void setImportStatementPool(String)
```

Default Value

No default value

Data Type

String

InitializationString

Purpose

Specifies one or multiple SQL commands to be executed by the driver after it has established a connection and has performed all initialization for the connection. If the execution of a SQL command fails, the connection attempt also fails and the driver throws an exception indicating which SQL command or commands failed.

Valid Values

command[[;*command*]...]

where:

command

is a SQL command.

Notes

Multiple commands must be separated by semicolons. In addition, if this property is specified in a connection URL, the entire value must be enclosed in parentheses when multiple commands are specified.

Data Source Methods

```
public String getInitializationString()
```

```
public void setInitializationString(String)
```

Default Value

No default value

Data Type

String

InsensitiveResultSetBufferSize

Purpose

Determines the amount of memory that is used by the driver to cache insensitive result set data.

Valid Values

`-1` | `0` | *x*

where:

x

is a positive integer that represents the amount of memory.

Behavior

If set to `-1`, the driver caches insensitive result set data in memory. If the size of the result set exceeds available memory, an OutOfMemoryException is generated. With no need to write result set data to disk, the driver processes the data efficiently.

If set to `0`, the driver caches insensitive result set data in memory, up to a maximum of 2 MB. If the size of the result set data exceeds available memory, then the driver pages the result set data to disk, which can have a negative performance effect. Because result set data may be written to disk, the driver may have to reformat the data to write it correctly to disk.

If set to *x*, the driver caches insensitive result set data in memory and uses this value to set the size (in KB) of the memory buffer for caching insensitive result set data. If the size of the result set data exceeds available memory, then the driver pages the result set data to disk, which can have a negative performance effect. Because the result set data may be written to disk, the driver may have to reformat the data to write it correctly to disk. Specifying a buffer size that is a power of 2 results in efficient memory use.

Data Source Methods

```
public Integer getInsensitiveResultsetBufferSize()
```

```
public void setInsensitiveResultsetBufferSize(Integer)
```

Default Value

2048

Data Type

Integer

See also

Performance considerations on page 45

KeywordConflictSuffix

Purpose

Specifies a string of up to 5 alphanumeric characters that the driver appends to any object or field name that conflicts with a SQL engine keyword.

Valid Values

String

where:

String

is a string of up to 5 alphanumeric characters.

Example

A field called CASE exists in the data schema. To avoid a naming conflict with the SQL engine keyword CASE, you could set KeywordConflictSuffix=TAB. In this scenario, the driver maps the CASE field to the CASETAB column.

Data Source Methods

```
public String getKeywordConflictSuffix()
```

```
public void setKeywordConflictSuffix(String)
```

Default Value

No default value

Data Type

String

LoginTimeout

Purpose

The amount of time, in seconds, that the driver waits for a connection to be established before timing out the connection request.

Valid Values

`-1` | `0` | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to `-1` | `0`, the driver does not time out a connection request.

If set to *x*, the driver waits for the specified number of seconds before returning control to the application and throwing a timeout exception.

Data Source Methods

```
public Integer getLoginTimeout()
```

```
public void setLoginTimeout(Integer)
```

Default Value

`0`

Data Type

Integer

MaxPooledStatements

Purpose

Specifies the maximum number of prepared statements to be pooled for each connection and enables the driver's internal prepared statement pooling when set to an integer greater than zero (0). The driver's internal prepared statement pooling provides performance benefits when the driver is not running from within an application server or another application that provides its own statement pooling.

Valid Values

`0` | *x*

where:

x

is a positive integer that represents a number of prepared statements to be cached.

Behavior

If set to `0`, the driver's internal prepared statement pooling is not enabled.

If set to *x*, the driver's internal prepared statement pooling is enabled and the driver uses the specified value to cache up to that many prepared statements created by an application. If the value set for this property is greater than the number of prepared statements that are used by the application, all prepared statements that are created by the application are cached. Because CallableStatement is a sub-class of PreparedStatement, CallableStatements also are cached.

Example

If the value of this property is set to `20`, the driver caches the last 20 prepared statements that are created by the application.

Notes

When you enable statement pooling, your applications can access the Statement Pool Monitor directly with DataDirect-specific methods. However, you can also enable the Statement Pool Monitor as a JMX MBean. To enable the Statement Pool Monitor as an MBean, statement pooling must be enabled with MaxPooledStatements and the Statement Pool Monitor MBean must be registered using the RegisterStatementPoolMonitorMBean connection property.

Data Source Methods

```
public Integer getMaxPooledStatements()
```

```
public void setMaxPooledStatements(Integer)
```

Default Value

`0`

Data Type

Integer

See also

Performance considerations on page 45

ProxyPort

Purpose

Specifies the port number where the proxy server is listening for HTTP or HTTPS requests for the first connection.

Valid Values

port

where:

port

is the port number on which the proxy server is listening. Contact your system administrator to obtain the correct port.

Data Source Methods

```
public Integer getProxyPort()
```

```
public void setProxyPort(Integer)
```

Default Value

`0` which means that the default value is determined by whether the value specified for the ProxyHost property is an HTTP or HTTPS URL.

For HTTP: `80`

For HTTPS: `443`

Data Type

Integer

See also

ProxyHost on page 67

ProxyHost

Purpose

Identifies a proxy server to use for the first connection.

Valid Values

server_name | *IP_address*

where:

server_name

is the name of the proxy server, which may be qualified with the domain name.

IP_address

is an IP address, specified in either IPv4 or IPv6 format, or a combination of the two.

Data Source Methods

```
public String getProxyHost()
```

```
public void setProxyHost(String)
```

Default Value

No default value

Data Type

String

See also

ProxyPort on page 66

ProxyPassword

Purpose

Specifies the password needed to connect to a proxy server for the first connection.

Valid Values

password

where:

password

is a valid password for that server. Contact your system administrator to obtain a valid password.

Data Source Methods

```
public String getProxyPassword()
public void setProxyPassword(String)
```

Default Value

No default value

Data Type

String

See also

ProxyUser on page 68

ProxyUser

Purpose

Specifies the user name needed to connect to a proxy server for the first connection.

Valid Values

user_name

where:

user_name

is a valid user ID for the proxy server.

Data Source Methods

```
public String getProxyUser()
```

```
public void setProxyUser(String)
```

Default Value

No default value

Data Type

String

See also

ProxyPassword on page 68

QueryTimeout

Purpose

Sets the default query timeout (in seconds) for all statements created by a connection.

Valid Values

`-1` | `0` | *x*

where:

x

is a number of seconds.

Behavior

If set to `-1`, the query timeout functionality is disabled. The driver silently ignores calls to the Statement.setQueryTimeout() method.

If set to `0`, the default query timeout is infinite (the query does not time out).

If set to *x*, the driver uses the value as the default timeout for any statement that is created by the connection. To override the default timeout value that is set by this property, call the Statement.setQueryTimeout() method to set a timeout value for a particular statement.

Data Source Methods

```
public Integer getQueryTimeout()
```

```
public void setQueryTimeout(Integer)
```

Default Value

`0`

Data Type

Integer

RefreshToken

Purpose

Specifies the refresh token used to either request a new access token or renew an expired access token for OAuth 2.0 implementations.

Important: The refresh token is a confidential value used to authenticate to the service. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the refresh token you have obtained from the Google Analytics service.

Notes

- See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getRefreshToken()
```

```
public void setRefreshToken(String)
```

Default Value

No default value

Data Type

String

See also

AccessToken on page 52
OAuth 2.0 authentication on page 40

RegisterStatementPoolMonitorMBean

Purpose

Registers the Statement Pool Monitor as a JMX MBean when statement pooling has been enabled with MaxPooledStatements. This allows you to manage statement pooling with standard JMX API calls and to use JMX-compliant tools, such as JConsole.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver registers an MBean for the statement pool monitor for each statement pool. This gives applications access to the Statement Pool Monitor through JMX when statement pooling is enabled.

If set to `false`, the driver does not register an MBean for the Statement Pool Monitor for any statement pool.

Notes

- Registering the MBean exports a reference to the Statement Pool Monitor. The exported reference can prevent garbage collection on connections if the connections are not properly closed. When garbage collection does not take place on these connections, out of memory errors can occur.
- For more information, refer to "Statement Pool Monitor" in the *Progress DataDirect for JDBC Drivers Reference*.

Data Source Methods

```
public Boolean getRegisterStatementPoolMonitorMbean()
```

```
public void setRegisterStatementPoolMonitorMbean(Boolean)
```

Default Value

`false`

Data Type

Boolean

SchemaMap

Purpose

Specifies either the name or the absolute path and name of the configuration file where the map of the database data model is written. The driver looks for this file when connecting to a database instance. If the file does not exist, the driver creates one.

Valid Values

`string`

where:

`string`

is either the name or the absolute path and name (including the `.config` extension) of the configuration file. For example, if specifying a value of:

- `ABC`, the driver either creates or looks for the configuration file `ABC` in the working directory of your application.
- `C:\\Users\\Default\\AppData\\Local\\Progress\\DataDirect\\<driver>_Schema\\abc@defcorp.com.config`, the driver either creates or looks for the configuration file `abc@defcorp.com.config` in the directory `C:\Users\Default\AppData\Local\Progress\DataDirect\<driver>_Schema`.

Notes

- When connecting to a database instance, the driver looks for the schema map configuration file. If the configuration file does not exist, the driver creates the schema map configuration file using the name and location you have provided. If you do not provide a name and location for the configuration file, the driver creates it using default values.
- The driver uses the path specified in this connection property to store additional internal files.
- You can refresh the internal files related to an existing view of your data by using the SQL extension Refresh Map. Refresh Map runs a discovery against your native data and updates your internal files accordingly.

Data Source Methods

```
public String getSchemaMap()
```

```
public void setSchemaMap(String)
```

Default Value

The default is determined by the environment. The driver attempts to create the files in a subdirectory of the first available directory in the following order:

- Windows
  - DD_HOME environment variable
  - dd.home system property
  - LOCALAPPDATA environment variable
  - APPDATA environment variable
  - user.home system property
- UNIX/Linux
  - DD_HOME environment variable
  - dd.home system property
  - user.home system property

For both Windows and UNIX/Linux, the file path takes the following format:

```
<available_location>/progress/datadirect/<driver>_schema/<user_name>.config
```

Data Type

String

See also

Refresh Map (EXT) on page 81

Scope

Purpose

Specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token.

Valid Values

String

where:

String

is a space-separated list of security scopes.

Data Source Methods

`public String getScope()`

`public void setScope(String)`

Default Value

`https://www.googleapis.com/auth/analytics.manage.users.readonly`
`https://www.googleapis.com/auth/analytics.readonly`

Data Type

String

See also

OAuth 2.0 authentication on page 40

ShowDeprecatedObjects

Purpose

Defines whether the driver exposes objects that Google Analytics has marked as deprecated. By default, the driver does not expose deprecated objects.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver includes deprecated objects in the relational model.

If set to `false`, the driver does not include deprecated objects in the relational model.

Example

If your application uses deprecated objects, set ShowDeprecatedObjects to `true` while you work on rewriting your queries and table definitions. Once the queries and table definitions are updated, change the setting back to `false`.

Data Source Methods

```
public Boolean getShowDeprecatedObjects()
```

```
public void setShowDeprecatedObjects(Boolean)
```

Default Value

`false`

Data Type

Boolean

See also

Introduction to the Google Analytics Data Model on page 103

ShowInternalTables

Purpose

Determines whether the driver exposes the Google Analytics `Data` table. The `Data` table is an internal Google Analytics table that contains all the analytics data the has been collected for a website.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver exposes the `Data` table.

If set to `false`, the driver does not expose the `Data` table.

Data Source Methods

```
public Boolean getShowInternalTables()
```

```
public void setShowInternalTables(Boolean)
```

Default Value

`false`

Data Type

Boolean

See also

Introduction to the Google Analytics Data Model on page 103

StmtCallLimit

Purpose

Specifies the maximum number of web service calls the driver can make when executing any single SQL statement or metadata query.

Valid Values

`0` | *x*

where:

x is a positive integer that defines the maximum number of web service calls up to 2147483647 the driver can make when executing any single SQL statement or metadata query.

Behavior

If set to `0`, there is no limit.

If set to *x*, the driver uses this value to set the maximum number of web service calls on a single connection that can be made when executing a SQL statement. This limit can be overridden by changing the `STMT_CALL_LIMIT` session attribute using the `ALTER SESSION` statement. For example, the following statement sets the statement call limit to 10 web service calls:

```
ALTER SESSION SET STMT_CALL_LIMIT=10
```

If the web service call limit is exceeded, the behavior of the driver depends on the value specified for the `StmtCallLimitBehavior` property.

Data Source Methods

```
public Integer getStmtCallLimit()
```

```
public void setStmtCallLimit(Integer)
```

Default Value

`0`

Data Type

Integer

See also

StmtCallLimitBehavior on page 75

StmtCallLimitBehavior

Purpose

Specifies the behavior of the driver when the maximum web service call limit specified by the StmtCallLimit property is exceeded.

Valid Values

`ReturnResults | ErrorAlways`

Behavior

If set to `ReturnResults`, the driver returns any partial results it received prior to the call limit being exceeded. The driver generates a warning that not all of the results were fetched.

If set to `ErrorAlways`, the driver generates an exception if the maximum web service call limit is exceeded.

Data Source Methods

```
public String getStmtCallLimitBehavior()
```

```
public void setStmtCallLimitBehavior(String)
```

Default Value

`ErrorAlways`

Data Type

String

See also

StmtCallLimit on page 75

SubtractTables

Purpose

Specifies a comma-separated list of tables that you do not want to expose in the relational view of your data.

Valid Values

`String`

where:

`String`

is a comma-separated list of table names.

Notes

SubtractTables can be used when you want to define your own tables instead of using the predefined tables. For example, `SubtractTables=adSense,adWords`.

Data Source Methods

```
public String getSubtractTables()
```

```
public void setSubtractTables(String)
```

Default Value

No default value

Data Type

String

See also

Introduction to the Google Analytics Data Model on page 103

TransactionMode

Purpose

Specifies how the driver handles manual transactions.

Valid Values

`NoTransactions` | `Ignore`

Behavior

If set to `NoTransactions`, the data source and the driver do not support transactions. Metadata indicates that the driver does not support transactions.

If set to `Ignore`, the data source does not support transactions and the driver always operates in auto-commit mode. Calls to set the driver to manual commit mode and to commit transactions are ignored. Calls to rollback a transaction cause the driver to throw an exception indicating that no transaction is started. Metadata indicates that the driver supports transactions and the ReadUncommitted transaction isolation level.

Data Source Methods

```
public String getTransactionMode()
```

```
public void setTransactionMode(String)
```

Default Value

`NoTransactions`

Data Type

String

5

Supported SQL statements and extensions

The driver provides support for the SQL statements and the SQL extensions described in this section. SQL extensions are denoted by an (EXT) in the topic title.

For details, see the following topics:

- Alter Session (EXT)
- Explain Plan
- Refresh Map (EXT)
- Select
- Subqueries
- SQL expressions

Alter Session (EXT)

Purpose

Changes various attributes of a local or remote session. A local session maintains the state of the overall connection. A remote session maintains the state that pertains to a particular remote data source connection.

Syntax

```
ALTER SESSION SET attribute_name=value
```

where:

attribute_name

specifies the name of the attribute to be changed. Attributes apply to either local or remote sessions.

value

specifies the value for that attribute.

The following table lists the local and remote session attributes, and provides descriptions of each.

Table 4: Alter Session Attributes

Attribute Name	Session Type	Description
Current_Schema	Local	Sets the current schema for the local session. The current schema is the schema used when an identifier in a SQL statement is unqualified. The string value must be the name of a schema visible in the local session. For example: `ALTER SESSION SET CURRENT_SCHEMA=GOOGLEANALYTICS`
Stmt_Call_Limit	Local	Sets the maximum number of Web service calls the driver can make in executing a statement. Setting the Stmt_Call_Limit attribute has the same effect as setting the Statement Call Limit connection option. It sets the default Web service call limit used by any statement on the connection. Executing this command on a statement overrides the previously set Statement Call Limit for the connection. The value specified must be a positive integer or `0`. The value `0` means that no call limit exists. For example: `ALTER SESSION SET STMT_CALL_LIMIT=150`

Ws_Call_Count	Remote	Resets the Web service call count of a remote session to the value specified. The value must be `0` or a positive integer. WS_Call_Count represents the total number of Web service calls made to the remote data source instance for the current session. For example:
		`ALTER SESSION SET googleanalytics.WS_CALL_COUNT=0`
		The current value of WS_Call_Count can be obtained by referring to the System_Remote_Sessions system table (see SYSTEM_REMOTE_SESSIONS Catalog Table for details). For example:
		`SELECT * from information_schema.system_remote_sessions WHERE session_id = cursessionid()`

Explain Plan

Purpose

Retrieves a detailed list of the elements in the execution plan. It generates a result set with a single column named `OPERATION`. The individual elements that comprise the plan are returned as rows in the result set.

Syntax

```
EXPLAIN PLAN FOR {SELECT ...}
```

The returned list of elements includes the indexes used for performing the query and can be used to optimize the query.

Refresh Map (EXT)

Purpose

The REFRESH MAP statement adds newly discovered objects to your relational view of native data. It also incorporates any configuration changes made to your relational view by reloading the schema definition and associated files.

Syntax

```
REFRESH MAP
```

Notes

- REFRESH MAP is an expensive query since it involves the discovery of native data.

Select

Purpose

Use the Select statement to fetch results from one or more tables.

Syntax

```
SELECT select_clause from_clause
[where_clause]
[groupby_clause]
[having_clause]
[{UNION [ALL | DISTINCT] |
 {MINUS [DISTINCT] | EXCEPT [DISTINCT]} |
 INTERSECT [DISTINCT]} select_statement]
[limit_clause]
```

where:

select_clause

specifies the columns from which results are to be returned by the query. See "Select clause" for a complete explanation.

from_clause

specifies one or more tables on which the other clauses in the query operate. See "From clause" for a complete explanation.

where_clause

is optional and restricts the results that are returned by the query. See "Where clause" for a complete explanation.

groupby_clause

is optional and allows query results to be aggregated in terms of groups. See "Group By clause" for a complete explanation.

having_clause

is optional and specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). See "Having clause" for a complete explanation.

UNION

is an optional operator that combines the results of the left and right Select statements into a single result. See "Union operator" for a complete explanation.

INTERSECT

is an optional operator that returns a single result by keeping any distinct values from the results of the left and right Select statements. See "Intersect operator" for a complete explanation.

EXCEPT | MINUS

are synonymous optional operators that returns a single result by taking the results of the left Select statement and removing the results of the right Select statement. See "Except and Minus operators" for a complete explanation.

orderby_clause

is optional and sorts the results that are returned by the query. See "Order By clause" for a complete explanation.

limit_clause

is optional and places an upper bound on the number of rows returned in the result. See "Limit clause" for a complete explanation.

Select clause

Purpose

Use the Select clause to specify with a list of column expressions that identify columns of values that you want to retrieve or an asterisk (*) to retrieve the value of all columns.

Syntax

```
SELECT [{LIMIT offsetnumber | TOP number}] [ALL | DISTINCT] {* | column_expression
[[AS] column_alias] [,column_expression [[AS] column_alias], ...]}
```

where:

`LIMIT offset number`

creates the result set for the Select statement first and then discards the first number of rows specified by *offset* and returns the number of remaining rows specified by *number*. To not discard any of the rows, specify `0` for *offset*, for example, `LIMIT 0` *number*. To discard the first *offset* number of rows and return all the remaining rows, specify `0` for *number*, for example, `LIMIT` *offset*`0`.

`TOP` *number*

is equivalent to `LIMIT 0`*number*.

column_expression

can be simply a column name (for example, `last_name`). More complex expressions may include mathematical operations or string manipulation (for example, `salary * 1.05`). See "SQL expressions" for details. *column_expression* can also include aggregate functions. See "Aggregate functions" for details.

column_alias

can be used to give the column a descriptive name. For example, to assign the alias department to the column dep:

```
SELECT dep AS department FROM emp
```

`DISTINCT`

eliminates duplicate rows from the result of a query. This operator can precede the first column expression. For example:

```
SELECT DISTINCT dep FROM emp
```

Notes

- Separate multiple column expressions with commas (for example, `SELECT last_name, first_name, hire_date`).
- Column names can be prefixed with the table name or table alias. For example, `SELECT emp.last_name or e.last_name`, where `e` is the alias for the table `emp`.
- NULL values are not treated as distinct from each other. The default behavior is that all result rows be returned, which can be made explicit with the keyword `ALL`.

See also

SQL expressions on page 94

Aggregate functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of rows. An aggregate can be used with a column name (for example, `AVG(salary)`) or in combination with a more complex column expression (for example, `AVG(salary * 1.07)`).

The following table lists supported aggregate functions.

Note: Doubly nested aggregates, such as `SUM(COUNT(col1))`, are currently not permitted by the driver.

Table 5: Aggregate Functions

Aggregate	Returns
AVG	The average of the values in a numeric column expression. For example, `AVG(salary)` returns the average of all salary column values.
COUNT	The number of values in any field expression. For example, `COUNT(name)` returns the number of name values. When using `COUNT` with a field name, `COUNT` returns the number of non-NULL column values. A special example is `COUNT(*)`, which returns the number of rows in the set, including rows with NULL values. **Note:** The driver does not support `COUNT(DISTINCT *)`. For example, `SELECT COUNT(DISTINCT *) FROM mytable` results in a syntax error.
MAX	The maximum value in any column expression. For example, `MAX(salary)` returns the maximum salary column value.
MIN	The minimum value in any column expression. For example, `MIN(salary)` returns the minimum salary column value.
SUM	The total of the values in a numeric column expression. For example, `SUM(salary)` returns the sum of all salary column values.

Example

The following example uses the `COUNT`, `MAX`, and `AVG` aggregate functions:

```
SELECT
     COUNT(amount) AS numOpportunities,
     MAX(amount) AS maxAmount,
     AVG(amount) AS avgAmount
FROM opportunity o INNER JOIN user u
     ON o.ownerId = u.id
WHERE o.isClosed = 'false' AND
     u.name = 'MyName'
```

From clause

Purpose

The From clause indicates the tables to be used in the Select statement.

Syntax

```
FROM table_name [table_alias] [,...]
```

where:

table_name

is the name of a table or a subquery. Multiple tables define an implicit inner join among those tables. Multiple table names must be separated by a comma. For example:

```
SELECT * FROM emp, dep
```

Subqueries can be used instead of table names. Subqueries must be enclosed in parentheses. See "Subquery in a From clause" for an example.

table_alias

is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias.

Example

The following example specifies two table aliases, `e` for `emp` and `d` for `dep`:

```
SELECT e.name, d.deptName
FROM emp e, dep d
WHERE e.deptId = d.id
```

table_alias is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias. For example, given the table specification:

```
FROM emp E
```

you may refer to the last_name field as E.last_name. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

Join in a From clause

Purpose

You can use a Join as a way to associate multiple tables within a Select statement. Joins may be either explicit or implicit. For example, the following is the example from the previous section restated as an explicit inner join:

```
SELECT * FROM emp INNER JOIN dep ON id=empId
SELECT e.name, d.deptName
FROM emp e INNER JOIN dep d ON e.deptId = d.id;
```

whereas the following is the same statement as an implicit inner join:

```
SELECT * FROM emp, dep WHERE emp.deptID=dep.id
```

Note: The `ON` clause in a join expression must evaluate to a true or false value.

Syntax

```
FROM table_name {RIGHT OUTER | INNER | LEFT OUTER | CROSS | FULL OUTER} JOIN table.key
 ON search-condition
```

Example

In this example, two tables are joined using `LEFT OUTER JOIN`. `T1`, the first table named includes nonmatching rows.

```
SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.key = T2.key
```

If you use a `CROSS JOIN`, no `ON` expression is allowed for the join.

Subquery in a From clause

Subqueries can be used in the From clause in place of table references (*`table_name`*).

Example

```
SELECT * FROM (SELECT * FROM emp WHERE sal > 10000) new_emp, dept WHERE
new_emp.deptno = dept.deptno
```

See also

Subqueries on page 92

Where clause

Purpose

Specifies the conditions that rows must meet to be retrieved.

Syntax

`WHERE` *`expr1 rel_operator expr2`*

where:

expr1

is either a column name, literal, or expression.

expr2

is either a column name, literal, expression, or subquery. Subqueries must be enclosed in parentheses.

rel_operator

is the relational operator that links the two expressions.

Example

The following Select statement retrieves the first and last names of employees that make at least $20,000.

```
SELECT last_name, first_name FROM emp WHERE salary >= 20000
```

See also

SQL expressions on page 94
Subqueries on page 92

Group By clause

Purpose

Specifies the names of one or more columns by which the returned values are grouped. This clause is used to return a set of aggregate values.

Syntax

```
GROUP BY column_expression [,...]
```

where:

column_expression

is either a column name or a SQL expression. Multiple values must be separated by a comma. If *column_expression* is a column name, it must match one of the column names specified in the Select clause. Also, the Group By clause must include all non-aggregate columns specified in the Select list.

Example

The following example totals the salaries in each department:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

See also

SQL expressions on page 94
Subqueries on page 92

Having clause

Purpose

Specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). This clause is valid only if you have already defined a Group By clause.

Syntax

```
HAVING expr1 rel_operator expr2
```

where:

expr1 | *expr2*

can be column names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause. See "SQL expressions" for details regarding SQL expressions.

rel_operator

is the relational operator that links the two expressions.

Example

The following example returns only the departments that have salaries totaling more than $200,000:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id HAVING sum(salary) > 200000
```

See also

SQL expressions on page 94
Subqueries on page 92

Union operator

Purpose

Combines the results of two Select statements into a single result. The single result is all the returned rows from both Select statements. By default, duplicate rows are not returned. To return duplicate rows, use the All keyword (`UNION ALL`).

Syntax

```
select_statement
UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT]} | INTERSECT
[DISTINCT]select_statement
```

Notes

- When using the Union operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

Intersect operator

Purpose

Intersect operator returns a single result set. The result set contains rows that are returned by both Select statements. Duplicates are returned unless the Distinct operator is added.

Syntax

```
select_statement
INTERSECT [DISTINCT]
select_statement
```

where:

`DISTINCT`

eliminates duplicate rows from the results.

Notes

- When using the Intersect operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
INTERSECT [DISTINCT]
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
INTERSECT
SELECT salary, last_name FROM raises
```

Except and Minus operators

Purpose

Return the rows from the left Select statement that are not included in the result of the right Select statement.

Syntax

```
select_statement
{EXCEPT [DISTINCT] | MINUS [DISTINCT]}
select_statement
```

where:

`DISTINCT`

eliminates duplicate rows from the results.

Notes

- When using one of these operators, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
EXCEPT
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
EXCEPT
SELECT salary, last_name FROM raises
```

Order By clause

Purpose

The Order By clause specifies how the rows are to be sorted.

Syntax

```
ORDER BY sort_expression [DESC | ASC] [,...]
```

where:

sort_expression

is either the name of a column, a column alias, a SQL expression, or the positioned number of the column or expression in the select list to use.

The default is to perform an ascending (ASC) sort.

Example

To sort by `last_name` and then by `first_name`, you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2,3
```

In the second example, `last_name` is the second item in the Select list, so `ORDER BY 2,3` sorts by `last_name` and then by `first_name`.

See also

SQL expressions on page 94

Limit clause

Purpose

Places an upper bound on the number of rows returned in the result.

Syntax

```
LIMIT number_of_rows [OFFSET offset_number]
```

where:

number_of_rows

specifies a maximum number of rows in the result. A negative number indicates no upper bound.

OFFSET

specifies how many rows to skip at the beginning of the result set. *offset_number* is the number of rows to skip.

Notes

- In a compound query, the Limit clause can appear only on the final Select statement. The limit is applied to the entire query, not to the individual Select statement to which it is attached.

Example

The following example returns a maximum of 20 rows.

```
SELECT last_name, first_name FROM emp WHERE salary > 20000 ORDER BY dept_idc LIMIT
20
```

Subqueries

A query is an operation that retrieves data from one or more tables or views. In this reference, a top-level query is called a Select statement, and a query nested within a Select statement is called a subquery.

A subquery is a query expression that appears in the body of another expression such as a Select, an Update, or a Delete statement. In the following example, the second Select statement is a subquery:

```
SELECT * FROM emp WHERE deptno IN (SELECT deptno FROM dept)
```

IN predicate

Purpose

The In predicate specifies a set of values against which to compare a result set. If the values are being compared against a subquery, only a single column result set is returned.

Syntax

value [NOT] IN (*value1, value2,...*)

OR

value [NOT] IN (*subquery*)

Example

```
SELECT * FROM emp WHERE deptno IN
(SELECT deptno FROM dept WHERE dname <> 'Sales')
```

EXISTS predicate

Purpose

The Exists predicate is true only if the cardinality of the subquery is greater than 0; otherwise, it is false.

Syntax

```
EXISTS (subquery)
```

Example

```
SELECT empno, ename, deptno FROM emp e WHERE EXISTS
(SELECT deptno FROM dept WHERE e.deptno = dept.deptno)
```

UNIQUE predicate

Purpose

The Unique predicate is used to determine whether duplicate rows exist in a virtual table (one returned from a subquery).

Syntax

```
UNIQUE (subquery)
```

Example

```
SELECT * FROM dept d WHERE UNIQUE
(SELECT deptno FROM emp e WHERE e.deptno = d.deptno)
```

Correlated subqueries

Purpose

A correlated subquery is a subquery that references a column from a table referred to in the parent statement. A correlated subquery is evaluated once for each row processed by the parent Select statement.

A correlated subquery answers a multiple-part question in which the answer depends on the value in each row processed by the parent statement. For example, you can use a correlated subquery to determine which employees earn more than the average salaries for their departments. In this case, the correlated subquery specifically computes the average salary for each department.

Syntax

```
SELECT select_list
   FROM table1 t_alias1
   WHERE expr rel_operator
   (SELECT column_list
       FROM table2 t_alias2
       WHERE t_alias1.columnrel_operatort_alias2.column)
```

Notes

- Correlated column names in correlated subqueries must be explicitly qualified with the table name of the parent.

Example A

The following statement returns data about employees whose salaries exceed their department average. This statement assigns an alias to emp, the table containing the salary information, and then uses the alias in a correlated subquery:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
  (SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
  ORDER BY deptno
```

Example B

This is an example of a correlated subquery that returns row values:

```
SELECT * FROM dept "outer" WHERE 'manager' IN
  (SELECT managername FROM emp
  WHERE "outer".deptno = emp.deptno)
```

Example C

This is an example of finding the department number (deptno) with multiple employees:

```
SELECT * FROM dept main WHERE 1 <
  (SELECT COUNT(*) FROM emp WHERE deptno = main.deptno)
```

Example D

This is an example of correlating a table with itself:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
  (SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
```

SQL expressions

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value. You can use expressions in the Where, and Having of Select statements; and in the Set clauses of Update statements.

Expressions enable you to use mathematical operations as well as character string manipulation operators to form complex queries.

The driver supports both unquoted and quoted identifiers. An unquoted identifier must start with an ASCII alpha character and can be followed by zero

Quoted identifiers must be enclosed in double quotation marks (""). A quoted identifier can contain any Unicode character including the space character. The driver recognizes the Unicode escape sequence \uxxxx as a Unicode character. You can specify a double quotation mark in a quoted identifier by escaping it with a double quotation mark.

The maximum length of both quoted and unquoted identifiers is 128 characters.

Valid expression elements are:

- Column names
- Literals
- Operators

- Functions

Column names

The most common expression is a simple column name. You can combine a column name with other expression elements.

Literals

Literals are fixed data values. For example, in the expression `PRICE * 1.05`, the value 1.05 is a constant. Literals are classified into types, including the following:

- Binary
- Character string
- Date
- Floating point
- Integer
- Numeric
- Time
- Timestamp

The following table describes the literal format for supported SQL data types.

Table 6: Literal Syntax Examples

SQL Type	Literal Syntax	Example
BIGINT	*n* where *n* is any valid integer value in the range of the INTEGER data type	`12 or -34 or 0`
BOOLEAN	Min Value: 0 Max Value: 1	`0` `1`
DATE	DATE'*date*'	`'2010-05-21'`
DATETIME	TIMESTAMP'*ts*'	`'2010-05-21 18:33:05.025'`

DECIMAL	*n.f*
	where:
	n
	is the integral part
	f
	is the fractional part
		`0.25`
		`3.1415`
		`-7.48`

SQL Type	Literal Syntax	Example
DOUBLE	*n.f*E*x* where: *n* is the integral part *f* is the fractional part *x* is the exponent	`1.2E0` or `2.5E40` or `-3.45E2` or `5.67E-4`
INTEGER	*n* where *n* is a valid integer value in the range of the INTEGER data type	`12` or `-34` or `0`
LONGVARBINARY	'*hex_value*'	`'000482ff'`
LONGVARCHAR	'*value*'	`'This is a string literal'`
TIME	TIME'*time*'	`'2010-05-21 18:33:05.025'`
VARCHAR	'*value*'	`'This is a string literal'`

Character string literals

Text specifies a character string literal. A character string literal must be enclosed in single quotation marks. To represent one single quotation mark within a literal, you must enter two single quotation marks. When the data in the fields is returned to the client, trailing blanks are stripped.

A character string literal can have a maximum length of 32 KB, that is, (32*1024) bytes.

Example

```
'Hello'
'Jim''s friend is Joe'
```

Numeric literals

Unquoted numeric values are treated as numeric literals. If the unquoted numeric value contains a decimal point or exponent, it is treated as a real literal; otherwise, it is treated as an integer literal.

Example

```
+1894.1204
```

Binary literals

Binary literals are represented with single quotation marks. The valid characters in a binary literal are 0-9, a-f, and A-F.

Example
```
'00af123d'
```

Date/Time literals

Date and time literal values are enclosed in single quotion marks ('*value*').

- The format for a Date literal is DATE'*date*'.
- The format for a Time literal is TIME'*time*'.
- The format for a Timestamp literal is TIMESTAMP'*ts*'.

Integer literals

Integer literals are represented by a string of numbers that are not enclosed in quotation marks and do not contain decimal points.

Notes

- Integer constants must be whole numbers; they cannot contain decimals.
- Integer literals can start with sign characters (+/-).

Example
```
1994 or -2
```

Operators

This section describes the operators that can be used in SQL expressions.

Note: Numeric operators are restricted to numeric types. Numeric operators do not support non-numeric types.

Unary operator

A unary operator operates on only one operand.

operator operand

Binary operator

A binary operator operates on two operands.

operand1 operator operand2

If an operator is given a null operand, the result is always null. The only operator that does not follow this rule is concatenation (||).

Arithmetic operators

You can use an arithmetic operator in an expression to negate, add, subtract, multiply, and divide numeric values. The result of this operation is also a numeric value. The + and - operators are also supported in date/time fields to allow date arithmetic. The following table lists the supported arithmetic operators.

Table 7: Arithmetic Operators

Operator	Purpose	Example
+ -	Denotes a positive or negative expression. These are unary operators.	`SELECT * FROM emp WHERE comm = -1`
* /	Multiplies, divides. These are binary operators.	`UPDATE emp SET sal = sal + sal * 0.10`
+ -	Adds, subtracts. These are binary operators.	`SELECT sal + comm FROM emp WHERE empno > 100`

Concatenation operator

The concatenation operator manipulates character strings. The following table lists the only supported concatenation operator.

Table 8: Concatenation Operator

Operator	Purpose	Example
||	Concatenates character strings.	`SELECT 'Name is' || ename FROM emp`

The result of concatenating two character strings is the data type VARCHAR.

Comparison operators

Comparison operators compare one expression to another. The result of such a comparison can be TRUE, FALSE, or UNKNOWN (if one of the operands is NULL). The driver considers the UNKNOWN result as FALSE.

The following table lists the supported comparison operators.

Table 9: Comparison Operators

Operator	Purpose	Example
=	Equality test.	`SELECT * FROM emp WHERE sal = 1500`
!=<>	Inequality test.	`SELECT * FROM emp WHERE sal != 1500`
><	"Greater than" and "less than" tests.	`SELECT * FROM emp WHERE sal > 1500 SELECT * FROM emp WHERE sal < 1500`
>=<=	"Greater than or equal to" and "less than or equal to" tests.	`SELECT * FROM emp WHERE sal >= 1500 SELECT * FROM emp WHERE sal <= 1500`

Operator	Purpose	Example
LIKE	% and _ wildcards can be used to search for a pattern in a column. The percent sign denotes zero, one, or multiple characters, while the underscore denotes a single character. The right-hand side of a LIKE expression must evaluate to a string or binary.	`SELECT * FROM emp WHERE ENAME LIKE 'J%'`

ESCAPE clause in LIKE operator
LIKE 'pattern string' ESCAPE 'c'
	The Escape clause is supported in the LIKE predicate to indicate the escape character. Escape characters are used in the pattern string to indicate that any wildcard character that is after the escape character in the pattern string should be treated as a regular character.
	The default escape character is backslash (\).
		`SELECT * FROM emp WHERE ENAME LIKE 'J%\_%' ESCAPE '\'`
		This matches all records with names that start with letter 'J' and have the '_' character in them.
		`SELECT * FROM emp WHERE ENAME LIKE 'JOE\_JOHN' ESCAPE '\'`
		This matches only records with name 'JOE_JOHN'.

[NOT] IN	"Equal to any member of" test.	`SELECT * FROM emp WHERE job IN ('CLERK','ANALYST') SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)`
[NOT] BETWEEN x AND y	"Greater than or equal to x" and "less than or equal to y."	`SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000`
EXISTS	Tests for existence of rows in a subquery.	`SELECT empno, ename, deptno FROM emp e WHERE EXISTS (SELECT deptno FROM dept WHERE e.deptno = dept.deptno)`
IS [NOT] NULL	Tests whether the value of the column or expression is NULL.	`SELECT * FROM emp WHERE ename IS NOT NULL SELECT * FROM emp WHERE ename IS NULL`

Logical operators

A logical operator combines the results of two component conditions to produce a single result or to invert the result of a single condition. The following table lists the supported logical operators.

Table 10: Logical Operators

Operator	Purpose	Example
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	`SELECT * FROM emp WHERE NOT (job IS NULL)` `SELECT * FROM emp WHERE NOT (sal BETWEEN 1000 AND 2000)`
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' AND deptno = 10`
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10`

Example

In the Where clause of the following Select statement, the AND logical operator is used to ensure that managers earning more than $1000 a month are returned in the result:

```
SELECT * FROM emp WHERE jobtitle = manager AND sal > 1000
```

Operator precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. The following table shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.You can change the order of precedence by using parentheses. Enclosing expressions in parentheses forces them to be evaluated together.

Table 11: Operator Precedence

Precedence	Operator
1	+ (Positive), - (Negative)
2	*(Multiply), / (Division)
3	+ (Add), - (Subtract)
4	|| (Concatenate)
5	=, >, <, >=, <=, <>, != (Comparison operators)
6	NOT, IN, LIKE
7	AND
8	OR

Example A

The query in the following example returns employee records for which the department number is 1 or 2 and the salary is greater than $1000:

```
SELECT * FROM emp WHERE (deptno = 1 OR deptno = 2) AND sal > 1000
```

Because parenthetical expressions are forced to be evaluated first, the OR operation takes precedence over AND.

Example B

In the following example, the query returns records for all the employees in department 1, but only employees whose salary is greater than $1000 in department 2.

```
SELECT * FROM emp WHERE deptno = 1 OR deptno = 2 AND sal > 1000
```

The AND operator takes precedence over OR, so that the search condition in the example is equivalent to the expression `deptno = 1 OR (deptno = 2 AND sal > 1000)`.

Functions

The driver supports a number of functions that you can use in expressions, including String, Numeric, Timedate, and System functions.

Refer to "Scalar functions" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

Conditions

A condition specifies a combination of one or more expressions and logical operators that evaluates to either TRUE, FALSE, or UNKNOWN. You can use a condition in the Where clause of the Delete, Select, and Update statements; and in the Having clauses of Select statements. The following describes supported conditions.

Table 12: Conditions

Condition	Description
Simple comparison	Specifies a comparison with expressions or subquery results. `= , != <>, < , >, <=, <=`
Group comparison	Specifies a comparison with any or all members in a list or subquery. `[= , != <>, < , >, <=, <=] [ANY, ALL, SOME]`
Membership	Tests for membership in a list or subquery. `[NOT] IN`
Range	Tests for inclusion in a range. `[NOT] BETWEEN`

Condition	Description
NULL	Tests for nulls. `IS NULL, IS NOT NULL`
EXISTS	Tests for existence of rows in a subquery. `[NOT] EXISTS`
LIKE	Specifies a test involving pattern matching. `[NOT] LIKE`
Compound	Specifies a combination of other conditions. `CONDITION [AND/OR] CONDITION`

6

Introduction to the Google Analytics Data Model

The driver exposes the Google Analytics data model in the form of a relational schema. Google Analytics stores two types of data: management data and analytics data. Management data consists of objects related to the management of Google Analytics services. The driver maps these objects directly as relational tables. Analytics data consists of all the data that Google Analytics collects about a website. This data is stored in a table named `Data`. Given the volume of information stored in the `Data` table, and the challenges in formulating useful queries against such a large data set, the driver hides the `Data` table by default. The driver provides thirteen Custom tables that contain compatible Google Analytics metrics and dimensions. In effect, custom tables allow you to simplify and write more effective queries against Google Analytics. Custom tables can be added to your Google Analytics relational schema, and further customized, using the Configuration Manager. Nevertheless, by setting the ShowInternalTables connection property to `true`, you can expose the `Data` table, allowing you to query it directly.

Note: The driver itself defines a number of functional tables, such as `INCOMPATIBILITY` and `METADATA`. The driver uses these tables to translate the Google Analytics data model into a relational schema. The metadata for these tables, as well as the management and custom tables, is provided in corresponding topics beginning with the ACCOUNT topic.

The following sections provide information to help you write effective queries against Google Analytics.

- Google Analytics data structure
- Custom table query
- Direct query
- Adding tables
- Defining columns

Google Analytics data structure

Google Analytics is a service that generates detailed statistics about a website's traffic and traffic sources. But Google Analytics is not just a database. It is a multi-dimensional hypercube containing all kinds of measurements about traffic to a website. When you connect to Google Analytics using the driver, you can reach into this repository and flatten it into relational data that can be used with your JDBC application.

Imagine a very small store of data about your website. For each hit, the Google Analytics service logs the date, language of user, country of origin, new or returning user, and their time on the site (in seconds).

2019-01-01	en	US	new	5.3
2019-01-01	nl	DK	new	90.4
2019-01-01	es	ES	new	24
2019-01-01	ja	JP	new	4.2
2019-01-01	es	MX	new	345.3
2019-01-01	ja	JP	returning	655.9
2019-01-02	en	US	new	45.7
2019-01-02	en	US	new	345.9
2019-01-02	es	ES	new	57.7
2019-01-02	en	US	new	6.8
2019-01-03	es	MX	new	876.1
2019-01-03	ja	JP	returning	5.7
2019-01-03	en	GB	new	5.6
2019-01-03	en	US	new	617.9
2019-01-03	en	US	returning	56.1
2019-01-04	es	MX	new	45.1
2019-01-04	jp	JP	new	178
2019-01-04	en	US	returning	103.9

Google Analytics collected data for this website over four days. The data is broken down by date, language, country and user type. For each visit, the time spent on the site was recorded. The time on the site is a *metric*, and the other columns are *dimensions*.

In an actual scenario, Google Analytics aggregates a vast amount of information about your website. It measures hundreds of things, and categorizes them by hundreds of dimensions. The query interface that Google Analytics provides allows you to fetch these metrics and group them. Because of the massive amount of information they store, their interface limits you to fetching up to ten metrics at a time, grouped by no more than seven dimensions.

For example, suppose you want to know how much time new visitors spent on the site. Your dimension is user type and your metric is time. You would get back two rows:

new	2648
returning	821.6

How much data you get back depends on how you ask for it. If you ask for two dimensions, you get even more data, because you get one row per permutation. Requesting how much time users have spent on each day, broken down by country, returns more rows:

2019-01-01	DK	90.4
2019-01-01	ES	24
2019-01-01	JP	660.1
2019-01-01	MX	345.3
2019-01-01	US	5.3
2019-01-02	ES	57.7
2019-01-02	US	398.4
2019-01-03	GB	5.6
2019-01-03	JP	5.7
2019-01-03	MX	876.1
2019-01-03	US	674
2019-01-04	JP	178
2019-01-04	MX	45.1
2019-01-04	US	103.9

Custom table query

The `Overview` table is one of the custom tables exposed by the driver. The `Overview` table exposes the Audience Overview information made available via the Google Analytics dashboard. Audience Overview has a graph showing Sessions, Users, Pageviews, Pages/Session, Average Session Duration, Bounce Rate and Percent of New Sessions. In the lower right, there is a breakdown of sessions by language.

Once connected to Google Analytics, you can query custom tables in one of two ways. First, you can formulate queries against custom tables by explicitly providing the Google Analytics View ID in each WHERE clause. For example:

```
SELECT * FROM Overview WHERE viewId = 'ga:12345678'
```

Note: A View ID can be obtained from your Google Analytics dashboard (Dashboard>Admin>View Settings>View ID).

Alternatively, you can specify the DefaultView connection property in your connection string. This avoids having to include the View ID in each WHERE clause. The value of DefaultView should be the value of the "View Name" field in the Google Analytics dashboard (Dashboard>Admin>View Settings>View Name). With the DefaultView property specified, the WHERE clause filter is not needed. For example:

```
SELECT * FROM Overview
```

Each of these queries provides the same results:

VIEWID	SEGMENTID	STARTDATE	ENDDATE	_BROWSER	_OPERATINGSYSTEM
ga:12345678	NULL	"2019-01-01"	"2019-01-30"	NULL	NULL

Note: To make the difference between metrics and dimensions clear, dimension names have an underscore prefix.

Important: Only one row was returned, and all of the dimensions came out as NULL because the driver enforces a rule that says if you ask for all dimensions, like we did with the SELECT *, then no dimensions are returned.

The DefaultQueryOptions connection property can also be used to filter results. DefaultQueryOptions specifies the values of Google Analytics `startDate`, `endDate`, and `viewId` query parameters for WHERE clause filtering during a session. Providing values for these parameters via DefaultQueryOptions simplifies queries and may produce more useful results.

Direct query

As discussed above, all the analytics data in Google Analytics resides in a hidden table named `Data`. If you want to query the `Data` table directly, you must expose the `Data` table by the ShowInternalTables connection property to `true`. With `ShowInternalTables=true`, you could execute the following query.

```
SELECT _LANGUAGE,SESSIONS FROM Data
```

In this case, the query against the `Data` table is functionally equivalent to the following custom table query.

```
SELECT _LANGUAGE,SESSIONS FROM Overview
```

However, it should be noted that hiding the `Data` table avoids queries such as `SELECT TOP 10 * FROM Data`, which is unlikely to yield useful results.

Adding tables

As discussed above, custom tables can be added to your Google Analytics relational schema, and further customized, using the Configuration Manager. Once you open the Configuration Manager, you can add and customize tables from the **Configure Logical Schema** dialog (Schema Settings>Add Tables>Configure Logical Schema). This dialog allows you define a new table, or customize an existing table, in terms of metrics and dimensions. You can select up to ten metrics and seven dimensions per table. Based on your selections, a table definition is created in the form of a JSON string which is incorporated into the connection string as a value for the AddTables property. For example:

```
AddTables={"MyTable":["sessions","_language","_country"]}
```

The table that results from this example would include the three selected columns, plus the columns `viewId`, `segmentId`, `startDate` and `endDate`. Now, as opposed to querying the `Data` table as we did above, we accomplish the functional equivalent by querying `MyTable`:

```
SELECT _LANGUAGE,SESSIONS FROM MyTable
```

Note: This table could be defined based on sessions and language alone, but the driver enforces a rule that says if you ask for all dimensions, then no dimensions are returned. This means that both `SELECT _LANGUAGE,SESSIONS` and `SELECT *` reference *one* dimension, and therefore, the data is not broken down by language. There is no harm in adding extra dimensions to your definition.

Note:

The SubtractTables connection property allows you to specify a comma-separated list of tables that you do not want to expose in the relational view of your data.

Defining the columns

The `Metadata` table can be used to define the columns in a table. The `Metadata` table has a list of all metrics and dimensions. Use only the metrics and dimensions that are marked with a "PUBLIC" status. By default, the driver ignores metrics and dimensions with a "DEPRECATED" status. You can expose deprecated metrics and dimensions by setting ShowDeprecatedObjects to `true`.

Not all combinations of metrics and dimensions are valid. Refer to the table called `Incompatibility`. If you see a row in that table that contains both columns, it means they can't be used in the same query.

For details, see the following topics:

- Custom tables
- ACCOUNT
- ACCOUNTSUMMARY
- ACCOUNTUSERLINK
- ADSENSE
- ADWORDS
- ADWORDSLINK
- ADWORDSLINKACCOUNT
- ADWORDSLINKPROFILE

- CAMPAIGN
- COMMERCE
- CUBE
- CUBEFIELD
- CUSTOMDATASOURCE
- CUSTOMDIMENSION
- CUSTOMMETRIC
- ENTRANCE
- EVENT
- EXIT
- EXPERIMENT
- EXPERIMENTVARIATION
- FILTER
- FLOODLIGHT
- GOAL
- INCOMPATIBILITY
- METADATA
- OAUTH
- OVERVIEW
- PAGESPEED
- PAGEUSAGE
- PROFILEFILTERLINK
- PROFILEUSERLINK
- SEARCH
- SEGMENT
- TABLE
- TEST
- UNSAMPLEDREPORT
- UPLOAD
- USAGE
- VIEW
- WEBPROPERTY
- WEBPROPERTYUSERLINK

Custom tables

The driver provides thirteen custom tables, as described in the table below. Each table contains compatible Google Analytics metrics and dimensions. Custom tables allow you to simplify and write more effective queries against Google Analytics. Custom tables can be added to your Google Analytics relational schema, and further customized, using the Configuration Manager.

Table	Dimensions Available	Metrics
Adsense	_campaign _date _day _dayOfWeek _dayOfWeekName _hour _month _week _year _yearMonth	adsenseAdsClicks adsensePageImpressions adsenseRevenue goalValueAll organicSearches sessions transactionRevenue users
Adwords	_campaign _date _day _dayOfWeek _dayOfWeekName _hour _month _week _year _yearMonth	adClicks adCost goalValueAll impressions organicSearches sessions transactionRevenue

Table	Dimensions Available	Metrics
Campaign	_date _day _dayOfWeek _dayOfWeekName _hour _month _week _year _yearMonth _dcmLastEventAd _dcmLastEventAdType _dcmLastEventAdvertiser _dcmLastEventCampaign _dcmLastEventCreative _dcmLastEventCreativeType _dcmLastEventCreativeVersion _dcmLastEventSite _dcmLastEventSitePlacement _dcmLastEventSpotId	dcmCPC dcmCTR dcmClicks dcmCost dcmImpressions dcmMargin dcmROI dcmRPC

Table	Dimensions Available	Metrics
Commerce	_affiliation _campaign _city _continent _country _date _day _dayOfWeek _dayOfWeekName _daysSinceLastSession _daysToTransaction _hostname _hour _keyword _language _metro _month _networkDomain _networkLocation _region _source _userGender _userType _week _year _yearMonth	itemQuantity sessions transactionRevenue transactionShipping transactionTax transactions uniquePurchases users

Table	Dimensions Available	Metrics
Entrance	_browser _city _continent _country _date _day _dayOfWeek _dayOfWeekName _hour _metro _month _networkDomain _networkLocation _region _week _year _yearMonth	bounces entranceRate entrances pageviews sessions users

Table	Dimensions Available	Metrics
Event	_city _continent _country _date _day _dayOfWeek _dayOfWeekName _eventAction _eventCategory _eventLabel _hour _language _metro _month _networkDomain _networkLocation _region _week _year	eventValue totalEvents uniqueEvents

Table	Dimensions Available	Metrics
Exit	_browser _city _continent _country _date _day _dayOfWeek _dayOfWeekName _hour _metro _month _networkDomain _networkLocation _region _week _year _yearMonth	bounces exitRate exits pageviews sessions users

Table	Dimensions Available	Metrics
Floodlight	_date _year _month _week _day _hour _dayOfWeek _dayOfWeekName _yearMonth _dcmClickAd _dcmClickAdvertiser _dcmClickCampaign _dcmClickCreative _dcmClickCreativeType _dcmClickCreativeVersion _dcmClickSite _dcmClickSitePlacement _dcmClickSpotId _dcmFloodlightActivity _dcmFloodlightActivityGroup	dcmFloodlightQuantity dcmFloodlightRevenue
Overview	_browser _city _country _date _language _networkLocation _operatingSystem _screenResolution	avgSessionDuration bounceRate pageviews pageviewsPerSession percentNewSessions sessions users

Table	Dimensions Available	Metrics
PageSpeed	_date _day _dayOfWeek _dayOfWeekName _hour _month _week _year _yearMonth	domainLookupTime pageDownloadTime pageLoadSample pageLoadTime pageviews redirectionTime serverConnectionTime serverResponseTime speedMetricsSample
PageUsage	_browser _campaign _city _continent _country _date _day _dayOfWeek _dayOfWeekName _hour _metro _month _networkDomain _networkLocation _operatingSystem _region _source _week _year _yearMonth	bounces goalCompletionsAll newUsers pageviews sessionDuration sessions transactionRevenue users

Table	Dimensions Available	Metrics
Search	_browser _city _continent _country _date _dayOfWeek _dayOfWeekName _language _metro _networkDomain _networkLocation _operatingSystem _region _searchCategory _searchDestinationPage _searchKeyword _searchStartPage _searchUsed	organicSearches searchDepth searchDuration searchExits searchUniques
Usage		bounces entrances exits pageLoadSample pageLoadTime pageviews sessionDuration sessions timeOnPage uniquePageviews

Table	Dimensions Available	Metrics
	_city _continent _country _date _day _dayOfWeek _dayOfWeekName _hostname _hour _language _metro _month _networkDomain _networkLocation _region _userType _week _year _yearMonth	

ACCOUNT

Columns

The ACCOUNT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
CHILDLINK_HREF	URL(255)
CHILDLINK_TYPE	String(255)
CREATED	Datetime(23)
NAME	String(255)
PERMISSIONS_EFFECTIVE	Array(255)

Column Name	Data Type
SELFLINK	URL(255)
UPDATED	Datetime(3)

ACCOUNTSUMMARY

Columns

The ACCOUNTSUMMARY table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROFILEID*	String(255)
ACCOUNTID	String(255)
ACCOUNTNAME	String(255)
INTERNALWEBPROPERTYID	String(255)
LEVEL	String(255)
NAME	String(255)
TYPE	String(255)
USERNAME	String(255)
WEBPROPERTYID	String(255)
WEBPROPERTYNAME	String(255)
WEBSITEURL	URL(255)

ACCOUNTUSERLINK

Columns

The ACCOUNTUSERLINK table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)

Column Name	Data Type
ACCOUNTHREF	URL(255)
ACCOUNTID	String(255)
ACCOUNTNAME	String(255)
INTERNALWEBPROPERTYID	String(255)
PERMISSIONS_EFFECTIVE	Array(255)
PERMISSIONS_LOCAL	Array(255)
PROFILEHREF	URL(255)
PROFILEID	String(255)
PROFILENAME	String(255)
SELFLINK	URL(255)
USEREMAIL	String(255)
USERID	String(255)
WEBPROPERTYHREF	URL(255)
WEBPROPERTYID	String(255)
WEBPROPERTYNAME	String(255)

ADSENSE

Columns

The ADSENSE view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_CAMPAIGN	String(255)	For manual campaign tracking, it is the value of the utm_campaign campaign tracking parameter. For AdWords autotagging, it is the name(s) of the online ad campaign(s) you use for the property. If you use neither, its value is (not set).
_DATE	Date	The date of the session formatted as YYYYMMDD.

Column Name	Data Type	Notes
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
ADSENSEADSCLICKS	Integer	The number of times AdSense ads on the site were clicked (requires integration with AdSense).
ADSENSEPAGEIMPRESSIONS	Integer	The number of pageviews during which an AdSense ad was displayed (requires integration with AdSense). A page impression can have multiple ad Units.
ADSENSEREVENUE	Decimal(18, 2)	The total revenue from AdSense ads.
ENDDATE	Date	
GOALVALUEALL	Decimal(18, 2)	Total numeric value for all goals defined in the profile.
ORGANICSEARCHES	Integer	The number of organic searches happened in a session. This metric is search engine agnostic.
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.

Column Name	Data Type	Notes
STARTDATE	Date	
TRANSACTIONREVENUE	Decimal(18, 2)	The total sale revenue (excluding shipping and tax) of the transaction.
USERS	Integer	The total number of users for the requested time period.
VIEWID	String(255)	

ADWORDS

Columns

The ADWORDS view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_CAMPAIGN	String(255)	For manual campaign tracking, it is the value of the utm_campaign campaign tracking parameter. For AdWords autotagging, it is the name(s) of the online ad campaign(s) you use for the property. If you use neither, its value is (not set).
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.

Column Name	Data Type	Notes
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
ADCLICKS	Integer	Total number of times users have clicked on an ad to reach the property.
ADCOST	Decimal(18, 2)	Derived cost for the advertising campaign. Its currency is the one you set in the AdWords account.
ENDDATE	Date	
GOALVALUEALL	Decimal(18, 2)	Total numeric value for all goals defined in the profile.
IMPRESSIONS	Integer	Total number of campaign impressions.
ORGANICSEARCHES	Integer	The number of organic searches happened in a session. This metric is search engine agnostic.
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
TRANSACTIONREVENUE	Decimal(18, 2)	The total sale revenue (excluding shipping and tax) of the transaction.
VIEWID	String(255)	

ADWORDSLINK

Columns

The ADWORDSLINK table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)

Column Name	Data Type
INTERNALWEBPROPERTYID	String(255)
NAME	String(255)
SELFLINK	URL(255)
WEBPROPERTYHREF	URL(255)
WEBPROPERTYID	String(255)
WEBPROPERTYNAME	String(255)

ADWORDSLINKACCOUNT

Columns

The ADWORDSLINKACCOUNT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CUSTOMERID*	String(255)
ACCOUNTID	String(255)
ADWORDSID	String(255)
AUTOTAGGINGENABLED	Boolean
WEBPROPERTYID	String(255)

ADWORDSLINKPROFILE

Columns

The ADWORDSLINKPROFILE table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROFILEID*	String(255)
ACCOUNTID	String(255)

Column Name	Data Type
ADWORDSID	String(255)
WEBPROPERTYID	String(255)

CAMPAIGN

Columns

The CAMPAIGN view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_DCMLASTEVENTAD	String(255)	DCM ad name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTADTYPE	String(255)	DCM ad type name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTADVERTISER	String(255)	DCM advertiser name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTCAMPAIGN	String(255)	DCM campaign name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTCREATIVE	String(255)	DCM creative name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).

Column Name	Data Type	Notes
_DCMLASTEVENTCREATIVETYPE	String(255)	DCM creative type name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTCREATIVEVERSION	String(255)	DCM creative version of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTSITE	String(255)	Site name where the DCM creative was shown and clicked on for the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTSITEPLACEMENT	String(255)	DCM site placement name of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_DCMLASTEVENTSPOTID	String(255)	DCM Floodlight configuration ID of the last DCM event (impression or click within the DCM lookback window) associated with the Google Analytics session (Analytics 360 only).
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
DCMCLICKS	Integer	DCM Total Clicks (Analytics 360 only).
DCMCOST	Decimal(18, 2)	DCM Total Cost (Analytics 360 only).

Column Name	Data Type	Notes
DCMCPC	Decimal(18, 2)	DCM Cost Per Click (Analytics 360 only).
DCMCTR	Percent	DCM Click Through Rate (Analytics 360 only).
DCMIMPRESSIONS	Integer	DCM Total Impressions (Analytics 360 only).
DCMRPC	Decimal(18, 2)	DCM Revenue Per Click (Analytics 360 only).
ENDDATE	Date	
SEGMENTID	String(255)	
STARTDATE	Date	
VIEWID	String(255)	

COMMERCE

Columns

The COMMERCE view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_AFFILIATION	String(255)	A product affiliation to designate a supplying company or brick and mortar location.
_CAMPAIGN	String(255)	For manual campaign tracking, it is the value of the utm_campaign campaign tracking parameter. For AdWords autotagging, it is the name(s) of the online ad campaign(s) you use for the property. If you use neither, its value is (not set).
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.

Column Name	Data Type	Notes
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_DAYSSINCELASTSESSION	String(255)	The number of days elapsed since users last visited the property, used to calculate user loyalty.
_DAYSTOTRANSACTION	String(255)	The number of days between users' purchases and the most recent campaign source prior to the purchase.
_HOSTNAME	String(255)	The hostname from which the tracking request was made.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_KEYWORD	String(255)	For manual campaign tracking, it is the value of the utm_term campaign tracking parameter. For AdWords traffic, it contains the best matching targeting criteria. For the display network, where multiple targeting criteria could have caused the ad to show up, it returns the best matching targeting criteria as selected by Ads. This could be display_keyword, site placement, boomuserlist, user_interest, age, or gender. Otherwise its value is (not set).
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.

Column Name	Data Type	Notes
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_SOURCE	String(255)	The source of referrals. For manual campaign tracking, it is the value of the utm_source campaign tracking parameter. For AdWords autotagging, it is google. If you use neither, it is the domain of the source (e.g., document.referrer) referring the users. It may also contain a port address. If users arrived without a referrer, its value is (direct).
_USERGENDER	String(255)	Gender of users.
_USERTYPE	String(255)	A boolean, either New Visitor or Returning Visitor, indicating if the users are new or returning.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
ENDDATE	Date	
ITEMQUANTITY	Integer	Total number of items purchased. For example, if users purchase 2 frisbees and 5 tennis balls, this will be 7.
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
TRANSACTIONREVENUE	Decimal(18, 2)	The total sale revenue (excluding shipping and tax) of the transaction.
TRANSACTIONS	Integer	The total number of transactions.
TRANSACTIONSHIPPING	Decimal(18, 2)	The total cost of shipping.
TRANSACTIONTAX	Decimal(18, 2)	Total tax for the transaction.

Column Name	Data Type	Notes
UNIQUEPURCHASES	Integer	The number of product sets purchased. For example, if users purchase 2 frisbees and 5 tennis balls from the site, this will be 2.
USERS	Integer	The total number of users for the requested time period.
VIEWID	String(255)	

CUBE

Columns

The CUBE table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
NAME	String(255)

CUBEFIELD

Columns

The CUBEFIELD table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
CUBEID	Integer
FIELD	String(255)

CUSTOMDATASOURCE

Columns

The CUSTOMDATASOURCE table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
CHILDLINK_HREF	URL(255)
CHILDLINK_TYPE	String(255)
CREATED	Datetime(3)
DESCRIPTION	String(255)
NAME	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
PROFILESLINKED	String(255)
SELFLINK	URL(255)
TYPE	CDSType(18)
UPDATED	Datetime(3)
WEBPROPERTYID	String(255)

CUSTOMDIMENSION

Columns

The CUSTOMDIMENSION table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
ACTIVE	Boolean
CREATED	Datetime(3)
INDEX	Integer
NAME	String(255)

Column Name	Data Type
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
SCOPE	String(255)
SELFLINK	URL(255)
UPDATED	Datetime(3)
WEBPROPERTYID	String(255)

CUSTOMMETRIC

Columns

The CUSTOMMETRIC table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
ACTIVE	Boolean
CREATED	Datetime(3)
INDEX	Integer
MAX_VALUE	String(255)
MIN_VALUE	String(255)
NAME	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
SCOPE	String(255)
SELFLINK	URL(255)
TYPE	String(255)

Column Name	Data Type
UPDATED	Datetime(3)
WEBPROPERTYID	String(255)

ENTRANCE

Columns

The ENTRANCE view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_BROWSER	String(255)	The name of users' browsers, for example, Internet Explorer or Firefox.
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.

Column Name	Data Type	Notes
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
BOUNCES	Integer	The total number of single page (or single interaction hit) sessions for the property.
ENDDATE	Date	
ENTRANCERATE	Percent	The percentage of pageviews in which this page was the entrance.
ENTRANCES	Integer	The number of entrances to the property measured as the first pageview in a session, typically used with landingPagePath.
PAGEVIEWS	Integer	The total number of pageviews for the property.
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
USERS	Integer	The total number of users for the requested time period.
VIEWID	String(255)	

EVENT

Columns

The EVENT view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_EVENTACTION	String(255)	Event action.
_EVENTCATEGORY	String(255)	The event category.
_EVENTLABEL	String(255)	Event label.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.

Column Name	Data Type	Notes
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
ENDDATE	Date	
EVENTVALUE	Integer	Total value of events for the profile.
SEGMENTID	String(255)	
STARTDATE	Date	
TOTALEVENTS	Integer	The total number of events for the profile, across all categories.
UNIQUEEVENTS	Integer	The number of unique events. Events in different sessions are counted as separate events.
VIEWID	String(255)	

EXIT

Columns

The EXIT view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_BROWSER	String(255)	The name of users' browsers, for example, Internet Explorer or Firefox.

Column Name	Data Type	Notes
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.

Column Name	Data Type	Notes
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
BOUNCES	Integer	The total number of single page (or single interaction hit) sessions for the property.
ENDDATE	Date	
EXITRATE	Percent	The percentage of exits from the property that occurred out of the total pageviews.
EXITS	Integer	The number of exits from the property.
PAGEVIEWS	Integer	The total number of pageviews for the property.
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
USERS	Integer	The total number of users for the requested time period.
VIEWID	String(255)	

EXPERIMENT

Columns

The EXPERIMENT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
CREATED	Datetime(3)
DESCRIPTION	String(255)
EDITABLEINGAUI	Boolean
ENDTIME	Datetime(3)

Column Name	Data Type
EQUALWEIGHTING	Boolean
INTERNALWEBPROPERTYID	String(255)
MINIMUMEXPERIMENTLENGTHINDAYS	Integer
NAME	String(255)
OBJECTIVEMETRIC	String(255)
OPTIMIZATIONTYPE	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
PROFILEID	String(255)
REASONEXPERIMENTENDED	String(255)
REWRITEVARIATIONURLSASORIGINAL	Boolean
SELFLINK	URL(255)
SERVINGFRAMEWORK	String(255)
SNIPPET	String(255)
STARTTIME	Datetime(3)
STATUS	String(255)
TRAFFICCOVERAGE	Double
UPDATED	Datetime(3)
WEBPROPERTYID	String(255)
WINNERCONFIDENCELEVEL	Double
WINNERFOUND	Boolean

EXPERIMENTVARIATION

Columns

The EXPERIMENTVARIATION table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
NAME*	String(255)
ACCOUNTID	String(255)
PROFILEID	String(255)
STATUS	String(255)
URL	URL(255)
WEBPROPERTYID	String(255)
WEIGHT	Double
WON	Boolean

FILTER

Columns

The FILTER table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
ADVANCEDDETAILS_CASESENSITIVE	Boolean
ADVANCEDDETAILS_EXTRACTA	String(255)
ADVANCEDDETAILS_EXTRACTB	String(255)
ADVANCEDDETAILS_FIELDA	String(255)
ADVANCEDDETAILS_FIELDAREQUIRED	Boolean
ADVANCEDDETAILS_FIELDB	String(255)
ADVANCEDDETAILS_FIELDBREQUIRED	Boolean
ADVANCEDDETAILS_OUTPUTCONSTRUCTOR	String(255)
ADVANCEDDETAILS_OUTPUTTOFIELD	String(255)

Column Name	Data Type
ADVANCEDDETAILS_OVERRIDEOUTPUTFIELD	Boolean
CREATED	Datetime(3)
EXCLUDEDETAILS_CASESENSITIVE	Boolean
EXCLUDEDETAILS_EXPRESSIONVALUE	String(255)
EXCLUDEDETAILS_FIELD	String(255)
EXCLUDEDETAILS_MATCHTYPE	String(255)
INCLUDEDETAILS_CASESENSITIVE	Boolean
INCLUDEDETAILS_EXPRESSIONVALUE	String(255)
INCLUDEDETAILS_FIELD	String(255)
INCLUDEDETAILS_MATCHTYPE	String(255)
LOWERCASEDETAILS_FIELD	String(255)
NAME	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
SEARCHANDREPLACEDETAILS_CASESENSITIVE	Boolean
SEARCHANDREPLACEDETAILS_FIELD	String(255)
SEARCHANDREPLACEDETAILS_REPLACESTRING	String(255)
SEARCHANDREPLACEDETAILS_SEARCHSTRING	String(255)
SELFLINK	URL(255)
TYPE	String(255)
UPDATED	Datetime(3)
UPPERCASEDETAILS_FIELD	String(255)

FLOODLIGHT

Columns

The FLOODLIGHT view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_DCMCLICKAD	String(255)	DCM ad name of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKADVERTISER	String(255)	DCM advertiser name of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKCAMPAIGN	String(255)	DCM campaign name of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKCREATIVE	String(255)	DCM creative name of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKCREATIVETYPE	String(255)	DCM creative type name of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKCREATIVEVERSION	String(255)	DCM creative version of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKSITE	String(255)	Site name where the DCM creative was shown and clicked on for the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKSITEPLACEMENT	String(255)	DCM site placement name of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMCLICKSPOTID	String(255)	DCM Floodlight configuration ID of the DCM click matching the Google Analytics session (Analytics 360 only).
_DCMFLOODLIGHTACTIVITY	String(255)	DCM Floodlight activity name associated with the floodlight conversion (Analytics 360 only).

Column Name	Data Type	Notes
_DCMFLOODLIGHTACTIVITYGROUP	String(255)	DCM Floodlight activity group name associated with the floodlight conversion (Analytics 360 only).
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
DCMFLOODLIGHTQUANTITY	Integer	The number of DCM Floodlight conversions (Analytics 360 only).
DCMFLOODLIGHTREVENUE	Decimal(18, 2)	DCM Floodlight revenue (Analytics 360 only).
ENDDATE	Date	
SEGMENTID	String(255)	
STARTDATE	Date	
VIEWID	String(255)	

GOAL

Columns

The GOAL table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)

Column Name	Data Type
ACCOUNTID	String(255)
ACTIVE	Boolean
CREATED	Datetime(3)
EVENTDETAILS_USEEVENTVALUE	Boolean
INTERNALWEBPROPERTYID	String(255)
NAME	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
PROFILEID	String(255)
SELFLINK	URL(255)
TYPE	String(255)
UPDATED	Datetime(3)
URLDESTINATIONDETAILS_CASESENSITIVE	Boolean
URLDESTINATIONDETAILS_FIRSTSTEPREQUIRED	Boolean
URLDESTINATIONDETAILS_MATCHTYPE	String(255)
URLDESTINATIONDETAILS_URL	URL(255)
VALUE	Float
VISITNUMPAGESDETAILS_COMPARISONTYPE	String(255)
VISITNUMPAGESDETAILS_COMPARISONVALUE	Long
VISITTIMEONSITEDETAILS_COMPARISONTYPE	String(255)
VISITTIMEONSITEDETAILS_COMPARISONVALUE	Long
WEBPROPERTYID	String(255)

INCOMPATIBILITY

Columns

The INCOMPATIBILITY table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
COLUMN1	String(255)
COLUMN2	String(255)

METADATA

Columns

The METADATA table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ALLOWEDINSEGMENTS	Boolean
CALCULATION	String(255)
DATATYPE	String(255)
DESCRIPTION	String(255)
GROUP	String(255)
MAXTEMPLATEINDEX	Integer
MINTEMPLATEINDEX	Integer
PREMIUMMAXTEMPLATEINDEX	Integer
PREMIUMMINTEMPLATEINDEX	Integer
REPLACEDBY	String(255)
STATUS	String(255)

Column Name	Data Type
TYPE	MetadataType(9)
UINAME	String(255)

OAUTH

Columns

The OAUTH table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
SCOPE	String(255)

OVERVIEW

Columns

The OVERVIEW view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_BROWSER	String(255)	The name of users' browsers, for example, Internet Explorer or Firefox.
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.

Column Name	Data Type	Notes
_OPERATINGSYSTEM	String(255)	Users' operating system, for example, Windows, Linux, Macintosh, or iOS.
_SCREENRESOLUTION	String(255)	Resolution of users' screens, for example, 1024x738.
AVGSESSIONDURATION	Time	The average duration (in seconds) of users' sessions.
BOUNCERATE	Percent	The percentage of single-page session (i.e., session in which the person left the property from the first page).
ENDDATE	Date	
PAGEVIEWS	Integer	The total number of pageviews for the property.
PAGEVIEWSPERSESSION	Float	The average number of pages viewed during a session, including repeated views of a single page.
PERCENTNEWSESSIONS	Percent	The percentage of sessions by users who had never visited the property before.
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
USERS	Integer	The total number of users for the requested time period.
VIEWID	String(255)	

PAGESPEED

Columns

The PAGESPEED view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.

Column Name	Data Type	Notes
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
DOMAINLOOKUPTIME	Integer	The total time (in milliseconds) all samples spent in DNS lookup for this page.
ENDDATE	Date	
PAGEDOWNLOADTIME	Integer	The total time (in milliseconds) to download this page among all samples.
PAGELOADSAMPLE	Integer	The sample set (or count) of pageviews used to calculate the average page load time.
PAGELOADTIME	Integer	Total time (in milliseconds), from pageview initiation (e.g., a click on a page link) to page load completion in the browser, the pages in the sample set take to load.
PAGEVIEWS	Integer	The total number of pageviews for the property.
REDIRECTIONTIME	Integer	The total time (in milliseconds) all samples spent in redirects before fetching this page. If there are no redirects, this is 0.
SEGMENTID	String(255)	

Column Name	Data Type	Notes
SERVERCONNECTIONTIME	Integer	Total time (in milliseconds) all samples spent in establishing a TCP connection to this page.
SERVERRESPONSETIME	Integer	The total time (in milliseconds) the site's server takes to respond to users' requests among all samples; this includes the network time from users' locations to the server.
SPEEDMETRICSSAMPLE	Integer	The sample set (or count) of pageviews used to calculate the averages of site speed metrics. This metric is used in all site speed average calculations, including avgDomainLookupTime, avgPageDownloadTime, avgRedirectionTime, avgServerConnectionTime, and avgServerResponseTime.
STARTDATE	Date	
VIEWID	String(255)	

PAGEUSAGE

Columns

The PAGEUSAGE view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_BROWSER	String(255)	The name of users' browsers, for example, Internet Explorer or Firefox.
_CAMPAIGN	String(255)	For manual campaign tracking, it is the value of the utm_campaign campaign tracking parameter. For AdWords autotagging, it is the name(s) of the online ad campaign(s) you use for the property. If you use neither, its value is (not set).
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.

Column Name	Data Type	Notes
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_OPERATINGSYSTEM	String(255)	Users' operating system, for example, Windows, Linux, Macintosh, or iOS.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_SOURCE	String(255)	The source of referrals. For manual campaign tracking, it is the value of the utm_source campaign tracking parameter. For AdWords autotagging, it is google. If you use neither, it is the domain of the source (e.g., document.referrer) referring the users. It may also contain a port address. If users arrived without a referrer, its value is (direct).
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.

Column Name	Data Type	Notes
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
BOUNCES	Integer	The total number of single page (or single interaction hit) sessions for the property.
ENDDATE	Date	
GOALCOMPLETIONSALL	Integer	Total number of completions for all goals defined in the profile.
NEWUSERS	Integer	The number of sessions marked as a user's first sessions.
PAGEVIEWS	Integer	The total number of pageviews for the property.
SEGMENTID	String(255)	
SESSIONDURATION	Time	Total duration (in seconds) of users' sessions.
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
TRANSACTIONREVENUE	Decimal(18, 2)	The total sale revenue (excluding shipping and tax) of the transaction.
USERS	Integer	The total number of users for the requested time period.
VIEWID	String(255)	

PROFILEFILTERLINK

Columns

The PROFILEFILTERLINK table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
FILTERHREF	URL(255)

Column Name	Data Type
FILTERID	String(255)
FILTERNAME	String(255)
INTERNALWEBPROPERTYID	String(255)
PROFILEHREF	URL(255)
PROFILEID	String(255)
PROFILENAME	String(255)
RANK	Integer
SELFLINK	URL(255)
WEBPROPERTYID	String(255)

PROFILEUSERLINK

Columns

The PROFILEUSERLINK table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTHREF	URL(255)
ACCOUNTID	String(255)
ACCOUNTNAME	String(255)
INTERNALWEBPROPERTYID	String(255)
PERMISSIONS_EFFECTIVE	Array(255)
PERMISSIONS_LOCAL	Array(255)
PROFILEHREF	URL(255)
PROFILEID	String(255)
PROFILENAME	String(255)
SELFLINK	URL(255)

Column Name	Data Type
USEREMAIL	String(255)
USERID	String(255)
WEBPROPERTYHREF	URL(255)
WEBPROPERTYID	String(255)
WEBPROPERTYNAME	String(255)

SEARCH

Columns

The SEARCH view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_BROWSER	String(255)	The name of users' browsers, for example, Internet Explorer or Firefox.
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.

Column Name	Data Type	Notes
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_OPERATINGSYSTEM	String(255)	Users' operating system, for example, Windows, Linux, Macintosh, or iOS.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_SEARCHCATEGORY	String(255)	The category used for the internal search if site search categories are enabled in the view. For example, the product category may be electronics, furniture, or clothing.
_SEARCHDESTINATIONPAGE	String(255)	The page users immediately visited after performing an internal search on the site. This is usually the search results page.
_SEARCHKEYWORD	String(255)	Search term used within the property.
_SEARCHSTARTPAGE	String(255)	The page where users initiated an internal search.
_SEARCHUSED	String(255)	A boolean, either Visits With Site Search or Visits Without Site Search, to distinguish whether internal search was used in a session.
ENDDATE	Date	
ORGANICSEARCHES	Integer	The number of organic searches happened in a session. This metric is search engine agnostic.
SEARCHDEPTH	Integer	The total number of subsequent page views made after a use of the site's internal search feature.
SEARCHDURATION	Time	The session duration when the site's internal search feature is used.
SEARCHEXITS	Integer	The number of exits on the site that occurred following a search result from the site's internal search feature.
SEARCHUNIQUES	Integer	Total number of unique keywords from internal searches within a session. For example, if "shoes" was searched for 3 times in a session, it would be counted only once.
SEGMENTID	String(255)	

Column Name	Data Type	Notes
STARTDATE	Date	
VIEWID	String(255)	

SEGMENT

Columns

The SEGMENT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
CREATED	Datetime(3)
DEFINITION	String(255)
NAME	String(255)
SEGMENTID	String(255)
SELFLINK	URL(255)
UPDATED	Datetime(3)

TABLE

Columns

The TABLE view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.
ENDDATE	Date	
SEGMENTID	String(255)	

Column Name	Data Type	Notes
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
VIEWID	String(255)	

TEST

Columns

The TEST view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.
ENDDATE	Date	
SEGMENTID	String(255)	
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
VIEWID	String(255)	

UNSAMPLEDREPORT

Columns

The UNSAMPLEDREPORT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	STRING(255)
ACCOUNTID	STRING(255)
CLOUDSTORAGEDOWNLOADDETAILS_BUCKETID	STRING(255)

Column Name	Data Type
CLOUDSTORAGEDOWNLOADDETAILS_OBJECTID	STRING(255)
CREATED	DATETIME(3)
DIMENSIONS	STRING(255)
DOWNLOADTYPE	STRING(255)
DRIVEDOWNLOADDETAILS_DOCUMENTID	STRING(255)
END-DATE	STRING(255)
FILTERS	STRING(255)
METRICS	STRING(255)
PROFILEID	STRING(255)
SEGMENT	STRING(255)
SELFLINK	URL(255)
START-DATE	STRING(255)
STATUS	STRING(255)
TITLE	STRING(255)
UPDATED	DATETIME(3)
WEBPROPERTYID	STRING(255)

UPLOAD

Columns

The UPLOAD table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
CUSTOMDATASOURCEID	String(255)
ERRORS	Array(255)

Column Name	Data Type
STATUS	String(255)
WEBPROPERTYID	String(255)

USAGE

Columns

The USAGE view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
_CITY	String(255)	Users' city, derived from their IP addresses or Geographical IDs.
_CONTINENT	String(255)	Users' continent, derived from users' IP addresses or Geographical IDs.
_COUNTRY	String(255)	Users' country, derived from their IP addresses or Geographical IDs.
_DATE	Date	The date of the session formatted as YYYYMMDD.
_DAY	String(255)	The day of the month, a two-digit number from 01 to 31.
_DAYOFWEEK	String(255)	Day of the week, a one-digit number from 0 (Sunday) to 6 (Saturday).
_DAYOFWEEKNAME	String(255)	Name (in English) of the day of the week.
_HOSTNAME	String(255)	The hostname from which the tracking request was made.
_HOUR	String(255)	A two-digit hour of the day ranging from 00-23 in the timezone configured for the account. This value is also corrected for daylight savings time. If the timezone follows daylight savings time, there will be an apparent bump in the number of sessions during the changeover hour (e.g., between 1:00 and 2:00) for the day per year when that hour repeats. A corresponding hour with zero sessions will occur at the opposite changeover. (Google Analytics does not track user time more precisely than hours.)
_LANGUAGE	String(255)	The language, in ISO-639 code format (e.g., en-gb for British English), provided by the HTTP Request for the browser.

Column Name	Data Type	Notes
_METRO	String(255)	The Designated Market Area (DMA) from where traffic arrived.
_MONTH	String(255)	Month of the session, a two digit integer from 01 to 12.
_NETWORKDOMAIN	String(255)	The domain name of users' ISP, derived from the domain name registered to the ISP's IP address.
_NETWORKLOCATION	String(255)	The names of the service providers used to reach the property. For example, if most users of the website come via the major cable internet service providers, its value will be these service providers' names.
_REGION	String(255)	Users' region, derived from their IP addresses or Geographical IDs. In U.S., a region is a state, New York, for example.
_USERTYPE	String(255)	A boolean, either New Visitor or Returning Visitor, indicating if the users are new or returning.
_WEEK	String(255)	The week of the session, a two-digit number from 01 to 53. Each week starts on Sunday.
_YEAR	String(255)	The year of the session, a four-digit year from 2005 to the current year.
_YEARMONTH	String(255)	Combined values of ga:year and ga:month.
BOUNCES	Integer	The total number of single page (or single interaction hit) sessions for the property.
ENDDATE	Date	
ENTRANCES	Integer	The number of entrances to the property measured as the first pageview in a session, typically used with landingPagePath.
EXITS	Integer	The number of exits from the property.
PAGELOADSAMPLE	Integer	The sample set (or count) of pageviews used to calculate the average page load time.
PAGELOADTIME	Integer	Total time (in milliseconds), from pageview initiation (e.g., a click on a page link) to page load completion in the browser, the pages in the sample set take to load.
PAGEVIEWS	Integer	The total number of pageviews for the property.
SEGMENTID	String(255)	

Column Name	Data Type	Notes
SESSIONDURATION	Time	Total duration (in seconds) of users' sessions.
SESSIONS	Integer	The total number of sessions.
STARTDATE	Date	
TIMEONPAGE	Time	Time (in seconds) users spent on a particular page, calculated by subtracting the initial view time for a particular page from the initial view time for a subsequent page. This metric does not apply to exit pages of the property.
UNIQUEPAGEVIEWS	Integer	Unique Pageviews is the number of sessions during which the specified page was viewed at least once. A unique pageview is counted for each page URL + page title combination.
VIEWID	String(255)	

VIEW

Columns

The VIEW table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
CHILDLINK_HREF	URL(255)
CHILDLINK_TYPE	String(255)
CREATED	Datetime(3)
CURRENCY	String(255)
DEFAULTPAGE	String(255)
ECOMMERCETRACKING	Boolean
ENHANCEDECOMMERCETRACKING	Boolean
EXCLUDEQUERYPARAMETERS	String(255)
INTERNALWEBPROPERTYID	String(255)

Column Name	Data Type
NAME	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
PERMISSIONS_EFFECTIVE	Array(255)
SELFLINK	URL(255)
SITESEARCHCATEGORYPARAMETERS	String(255)
SITESEARCHQUERYPARAMETERS	String(255)
STRIPSITESEARCHCATEGORYPARAMETERS	Boolean
STRIPSITESEARCHQUERYPARAMETERS	Boolean
TIMEZONE	String(255)
TYPE	String(255)
UPDATED	Datetime(3)
WEBPROPERTYID	String(255)
WEBSITEURL	URL(255)

WEBPROPERTY

Columns

The WEBPROPERTY table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTID	String(255)
CHILDLINK_HREF	URL(255)
CHILDLINK_TYPE	String(255)
CREATED	Datetime(3)
DEFAULTPROFILEID	String(255)

Column Name	Data Type
INDUSTRYVERTICAL	String(255)
INTERNALWEBPROPERTYID	String(255)
LEVEL	String(255)
NAME	String(255)
PARENTLINK_HREF	URL(255)
PARENTLINK_TYPE	String(255)
PERMISSIONS_EFFECTIVE	Array(255)
PROFILECOUNT	Integer
SELFLINK	URL(255)
UPDATED	Datetime(3)
WEBSITEURL	URL(255)

WEBPROPERTYUSERLINK

Columns

The WEBPROPERTYUSERLINK table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	String(255)
ACCOUNTHREF	URL(255)
ACCOUNTID	String(255)
ACCOUNTNAME	String(255)
INTERNALWEBPROPERTYID	String(255)
PERMISSIONS_EFFECTIVE	Array(255)
PERMISSIONS_LOCAL	Array(255)
PROFILEHREF	URL(255)
PROFILEID	String(255)

Column Name	Data Type
PROFILENAME	String(255)
SELFLINK	URL(255)
USEREMAIL	String(255)
USERID	String(255)
WEBPROPERTYHREF	URL(255)
WEBPROPERTYID	String(255)
WEBPROPERTYNAME	String(255)