Package ‘sparkbq’

October 14, 2022

Type Package

Title Google 'BigQuery' Support for 'sparklyr'

Version 0.1.1

URL `http://www.mirai-solutions.com`,
`https://github.com/miraisolutions/sparkbq`

BugReports `https://github.com/miraisolutions/sparkbq/issues`

Description A 'sparklyr' extension package providing an integration with Google 'BigQuery'.
It supports direct import/export where records are directly streamed from/to 'BigQuery'.
In addition, data may be imported/exported via intermediate data extracts on Google 'Cloud Storage'.

Depends R (>= 3.3.2)

Imports sparklyr (>= 0.7.0)

Suggests dplyr

License GPL-3 | file LICENSE

SystemRequirements Spark (>= 2.2.x)

Encoding UTF-8

LazyData yes

RoxygenNote 6.1.1

NeedsCompilation no

Author Mirai Solutions GmbH [aut],
Martin Studer [cre],
Nicola Lambiase [ctb],
Omer Demirel [ctb]

Maintainer Martin Studer <`martin.studer@mirai-solutions.com`>

Repository CRAN

Date/Publication 2019-12-18 18:00:02 UTC

R topics documented:

bigquery_defaults *Google BigQuery Default Settings*

Description

Sets default values for several Google BigQuery related settings.

Usage

```
bigquery_defaults(billingProjectId, gcsBucket, datasetLocation = "US",
  serviceAccountKeyFile = NULL, type = "direct")
```

Arguments

`billingProjectId`
: Default Google Cloud Platform project ID for billing purposes. This is the project on whose behalf to perform BigQuery operations.

`gcsBucket`
: Google Cloud Storage (GCS) bucket to use for storing temporary files. Temporary files are used when importing through BigQuery load jobs and exporting through BigQuery extraction jobs (i.e. when using data extracts such as Parquet, Avro, ORC, ...). The service account specified in `serviceAccountKeyFile` needs to be given appropriate rights. This should be the name of an existing storage bucket.

`datasetLocation`
: Geographic location where newly created datasets should reside. "EU" or "US". Defaults to "US".

`serviceAccountKeyFile`
: Google Cloud service account key file to use for authentication with Google Cloud services. The use of service accounts is highly recommended. Specifically, the service account will be used to interact with BigQuery and Google Cloud Storage (GCS). If not specified, Google application default credentials (ADC) will be used, which is the default.

`type` Default BigQuery import/export type to use. Options include "direct", "parquet", "avro", "orc", "json" and "csv". Defaults to "direct". Please note that only "direct" and "avro" are supported for both importing and exporting.

"csv" and "json" are not recommended due to their lack of type safety.

See the table below for supported type and import/export combinations.

	Direct	Parquet	Avro	ORC	JSON	CSV
Import to Spark (export from BigQuery)	X		X		X	X
Export from Spark (import to BigQuery)	X	X	X	X		

Value

A `list` of set options with previous values.

References

https://github.com/miraisolutions/spark-bigquery https://cloud.google.com/bigquery/pricing https://cloud.google.com/bigquery/docs/dataset-locations https://cloud.google.com/bigquery/docs/authentication/service-account-file https://cloud.google.com/docs/authentication/ https://cloud.google.com/bigquery/docs/authentication/ https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-parquet https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-avro https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-orc https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-json https://cloud.google.com/bigquery/docs/loading-data-cloud-stor

See Also

`spark_read_bigquery`, `spark_write_bigquery`, `default_billing_project_id`, `default_gcs_bucket`, `default_dataset_location`

`default_bigquery_type` *Default BigQuery import/export type*

Description

Returns the default BigQuery import/export type. It defaults to "direct".

Usage

```
default_bigquery_type()
```

See Also

`bigquery_defaults`

default_billing_project_id

Default Google BigQuery Billing Project ID

Description

Returns the default Google BigQuery billing project ID.

Usage

```
default_billing_project_id()
```

See Also

bigquery_defaults

default_dataset_location

Default Google BigQuery Dataset Location

Description

Returns the default Google BigQuery dataset location. It defaults to "US".

Usage

```
default_dataset_location()
```

References

https://cloud.google.com/bigquery/docs/dataset-locations

See Also

bigquery_defaults

`default_gcs_bucket` *Default Google BigQuery GCS Bucket*

Description

Returns the default Google BigQuery GCS bucket.

Usage

```
default_gcs_bucket()
```

See Also

`bigquery_defaults`

`default_service_account_key_file` *Default Google BigQuery Service Account Key File*

Description

Returns the default service account key file to use.

Usage

```
default_service_account_key_file()
```

References

`https://cloud.google.com/bigquery/docs/authentication/service-account-file` `https://cloud.google.com/docs/authentication/` `https://cloud.google.com/bigquery/docs/authentication/`

See Also

`bigquery_defaults`

spark_read_bigquery *Reading data from Google BigQuery*

Description

This function reads data stored in a Google BigQuery table.

Usage

```
spark_read_bigquery(sc, name,
  billingProjectId = default_billing_project_id(),
  projectId = billingProjectId, datasetId = NULL, tableId = NULL,
  sqlQuery = NULL, type = default_bigquery_type(),
  gcsBucket = default_gcs_bucket(),
  serviceAccountKeyFile = default_service_account_key_file(),
  additionalParameters = NULL, memory = FALSE, ...)
```

Arguments

`sc` `spark_connection` provided by sparklyr.

`name` The name to assign to the newly generated table (see also `spark_read_source`).

`billingProjectId` Google Cloud Platform project ID for billing purposes. This is the project on whose behalf to perform BigQuery operations. Defaults to `default_billing_project_id()`.

`projectId` Google Cloud Platform project ID of BigQuery dataset. Defaults to `billingProjectId`.

`datasetId` Google BigQuery dataset ID (may contain letters, numbers and underscores). Either both of `datasetId` and `tableId` or `sqlQuery` must be specified.

`tableId` Google BigQuery table ID (may contain letters, numbers and underscores). Either both of `datasetId` and `tableId` or `sqlQuery` must be specified.

`sqlQuery` Google BigQuery SQL query. Either both of `datasetId` and `tableId` or `sqlQuery` must be specified. The query must be specified in standard SQL (SQL-2011). Legacy SQL is not supported. Tables are specified as '<project_id>.<dataset_id>.<table_id>'.

`type` BigQuery import type to use. Options include "direct", "avro", "json" and "csv". Defaults to `default_bigquery_type()`. See bigquery_defaults for more details about the supported types.

`gcsBucket` Google Cloud Storage (GCS) bucket to use for storing temporary files. Temporary files are used when importing through BigQuery load jobs and exporting through BigQuery extraction jobs (i.e. when using data extracts such as Parquet, Avro, ORC, ...). The service account specified in `serviceAccountKeyFile` needs to be given appropriate rights. This should be the name of an existing storage bucket.

`serviceAccountKeyFile` Google Cloud service account key file to use for authentication with Google Cloud services. The use of service accounts is highly recommended. Specifically, the service account will be used to interact with BigQuery and Google Cloud Storage (GCS).

`additionalParameters`
Additional spark-bigquery options. See https://github.com/miraisolutions/spark-bigquery for more information.

`memory`
logical specifying whether data should be loaded eagerly into memory, i.e. whether the table should be cached. Note that eagerly caching prevents predicate pushdown (e.g. in conjunction with `filter`) and therefore the default is `FALSE`. See also `spark_read_source`.

`...`
Additional arguments passed to `spark_read_source`.

Value

A `tbl_spark` which provides a `dplyr`-compatible reference to a Spark DataFrame.

References

https://github.com/miraisolutions/spark-bigquery https://cloud.google.com/bigquery/docs/datasets https://cloud.google.com/bigquery/docs/tables https://cloud.google.com/bigquery/docs/reference/standard-sql/ https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-avro https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-j
https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-csv https://cloud.google.com/bigquery/pricing https://cloud.google.com/bigquery/docs/dataset-locations
https://cloud.google.com/docs/authentication/ https://cloud.google.com/bigquery/docs/authentication/

See Also

`spark_read_source`, `spark_write_bigquery`, `bigquery_defaults`

Other Spark serialization routines: `spark_write_bigquery`

Examples

```
## Not run:
config <- spark_config()

sc <- spark_connect(master = "local", config = config)

bigquery_defaults(
  billingProjectId = "<your_billing_project_id>",
  gcsBucket = "<your_gcs_bucket>",
  datasetLocation = "US",
  serviceAccountKeyFile = "<your_service_account_key_file>",
  type = "direct")

# Reading the public shakespeare data table
# https://cloud.google.com/bigquery/public-data/
# https://cloud.google.com/bigquery/sample-tables
shakespeare <-
  spark_read_bigquery(
    sc,
    name = "shakespeare",
    projectId = "bigquery-public-data",
    datasetId = "samples",
    tableId = "shakespeare")

## End(Not run)
```

`spark_write_bigquery` *Writing data to Google BigQuery*

Description

This function writes data to a Google BigQuery table.

Usage

```
spark_write_bigquery(data,
  billingProjectId = default_billing_project_id(),
  projectId = billingProjectId, datasetId, tableId,
  type = default_bigquery_type(), gcsBucket = default_gcs_bucket(),
  datasetLocation = default_dataset_location(),
  serviceAccountKeyFile = default_service_account_key_file(),
  additionalParameters = NULL, mode = "error", ...)
```

Arguments

`data` Spark DataFrame to write to Google BigQuery.

`billingProjectId` Google Cloud Platform project ID for billing purposes. This is the project on whose behalf to perform BigQuery operations. Defaults to `default_billing_project_id()`.

`projectId` Google Cloud Platform project ID of BigQuery dataset. Defaults to `billingProjectId`.

`datasetId` Google BigQuery dataset ID (may contain letters, numbers and underscores).

`tableId` Google BigQuery table ID (may contain letters, numbers and underscores).

`type` BigQuery export type to use. Options include "direct", "parquet", "avro", "orc". Defaults to `default_bigquery_type()`. See bigquery_defaults for more details about the supported types.

`gcsBucket` Google Cloud Storage (GCS) bucket to use for storing temporary files. Temporary files are used when importing through BigQuery load jobs and exporting through BigQuery extraction jobs (i.e. when using data extracts such as Parquet, Avro, ORC, ...). The service account specified in `serviceAccountKeyFile` needs to be given appropriate rights. This should be the name of an existing storage bucket.

`datasetLocation` Geographic location where newly created datasets should reside. "EU" or "US". Defaults to "US". Only needs to be specified if the dataset does not yet exist. It is ignored if it is specified and the dataset already exists.

`serviceAccountKeyFile`
: Google Cloud service account key file to use for authentication with Google Cloud services. The use of service accounts is highly recommended. Specifically, the service account will be used to interact with BigQuery and Google Cloud Storage (GCS).

`additionalParameters`
: Additional spark-bigquery options. See https://github.com/miraisolutions/spark-bigquery for more information.

`mode`
: Specifies the behavior when data or table already exist. One of "overwrite", "append", "ignore" or "error" (default).

`...`
: Additional arguments passed to `spark_write_source`.

Value

`NULL`. This is a side-effecting function.

References

https://github.com/miraisolutions/spark-bigquery https://cloud.google.com/bigquery/docs/datasets https://cloud.google.com/bigquery/docs/tables https://cloud.google.com/bigquery/docs/reference/standard-sql/ https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-parquet https://cloud.google.com/bigquery/docs/loading-data-cloud-storag https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-orc https://cloud.google.com/bigquery/pricing https://cloud.google.com/bigquery/docs/dataset-locations https://cloud.google.com/docs/authentication/ https://cloud.google.com/bigquery/docs/authentication/

See Also

`spark_write_source`, `spark_read_bigquery`, `bigquery_defaults`

Other Spark serialization routines: `spark_read_bigquery`

Examples

```
## Not run:
config <- spark_config()

sc <- spark_connect(master = "local", config = config)

bigquery_defaults(
  billingProjectId = "<your_billing_project_id>",
  gcsBucket = "<your_gcs_bucket>",
  datasetLocation = "US",
  serviceAccountKeyFile = "<your_service_account_key_file>",
  type = "direct")

# Copy mtcars to Spark
spark_mtcars <- dplyr::copy_to(sc, mtcars, "spark_mtcars", overwrite = TRUE)

spark_write_bigquery(
  data = spark_mtcars,
  datasetId = "<your_dataset_id>",
  tableId = "mtcars",
  mode = "overwrite")

## End(Not run)
```

Index