tf.contrib.cloud.BigQueryReader

Contents

Class **BigQueryReader**

Inherits From: **ReaderBase**

Defined in **tensorflow/contrib/cloud/python/ops/bigquery_reader_ops.py** .

A Reader that outputs keys and tf.Example values from a BigQuery table.

Example use:

```
# Assume a BigQuery has the following schema,
#     name      STRING,
#     age       INT,
#     state     STRING

# Create the parse_examples list of features.
features = dict(
  name=tf.FixedLenFeature([1], tf.string),
  age=tf.FixedLenFeature([1], tf.int32),
  state=tf.FixedLenFeature([1], dtype=tf.string, default_value="UNK"))

# Create a Reader.
reader = bigquery_reader_ops.BigQueryReader(project_id=PROJECT,
                                            dataset_id=DATASET,
                                            table_id=TABLE,
                                            timestamp_millis=TIME,
                                            num_partitions=NUM_PARTITIONS,
                                            features=features)

# Populate a queue with the BigQuery Table partitions.
queue = tf.train.string_input_producer(reader.partitions())

# Read and parse examples.
row_id, examples_serialized = reader.read(queue)
examples = tf.parse_example(examples_serialized, features=features)

# Process the Tensors examples["name"], examples["age"], etc...
```

Note that to create a reader a snapshot timestamp is necessary. This will enable the reader to look at a consistent snapshot of the table. For more information, see 'Table Decorators' in BigQuery docs.

See ReaderBase for supported methods.

Properties

`reader_ref`

Op that implements the reader.

`supports_serialize`

Whether the Reader implementation can serialize its state.

Methods

`__init__`

```
__init__(
    project_id,
    dataset_id,
    table_id,
    timestamp_millis,
    num_partitions,
    features=None,
    columns=None,
    test_end_point=None,
    name=None
)
```

Creates a BigQueryReader.

Args:

- `project_id` : GCP project ID.
- `dataset_id` : BigQuery dataset ID.
- `table_id` : BigQuery table ID.
- `timestamp_millis` : timestamp to snapshot the table in milliseconds since the epoch. Relative (negative or zero) snapshot times are not allowed. For more details, see 'Table Decorators' in BigQuery docs.
- `num_partitions` : Number of non-overlapping partitions to read from.
- `features` : parse_example compatible dict from keys to `VarLenFeature` and `FixedLenFeature` objects. Keys are read as columns from the db.
- `columns` : list of columns to read, can be set iff features is None.
- `test_end_point` : Used only for testing purposes (optional).
- `name` : a name for the operation (optional).

Raises:

- `TypeError` : - If features is neither None nor a dict or - If columns is neither None nor a list or - If both features and columns are None or set.

`num_records_produced`

```
num_records_produced(name=None)
```

Returns the number of records this reader has produced.

This is the same as the number of Read executions that have succeeded.

Args:

- `name`: A name for the operation (optional).

Returns:

An int64 Tensor.

`num_work_units_completed`

```
num_work_units_completed(name=None)
```

Returns the number of work units this reader has finished processing.

Args:

- `name`: A name for the operation (optional).

Returns:

An int64 Tensor.

`partitions`

```
partitions(name=None)
```

Returns serialized BigQueryTablePartition messages.

These messages represent a non-overlapping division of a table for a bulk read.

Args:

- `name`: a name for the operation (optional).

Returns:

`1-D` string `Tensor` of serialized `BigQueryTablePartition` messages.

`read`

```
read(
    queue,
    name=None
)
```

Returns the next record (key, value) pair produced by a reader.

Will dequeue a work unit from queue if necessary (e.g. when the Reader needs to start reading from a new file since it has finished with the previous file).

Args:

- `queue` : A Queue or a mutable string Tensor representing a handle to a Queue, with string work items.
- `name` : A name for the operation (optional).

Returns:

A tuple of Tensors (key, value). `key` *: A string scalar Tensor.* `value` : A string scalar Tensor.

`read_up_to`

```
read_up_to(
    queue,
    num_records,
    name=None
)
```

Returns up to num_records (key, value) pairs produced by a reader.

Will dequeue a work unit from queue if necessary (e.g., when the Reader needs to start reading from a new file since it has finished with the previous file). It may return less than num_records even before the last batch.

Args:

- `queue` : A Queue or a mutable string Tensor representing a handle to a Queue, with string work items.
- `num_records` : Number of records to read.
- `name` : A name for the operation (optional).

Returns:

A tuple of Tensors (keys, values). `keys` *: A 1-D string Tensor.* `values` : A 1-D string Tensor.

`reset`

```
reset(name=None)
```

Restore a reader to its initial clean state.

Args:

- `name` : A name for the operation (optional).

Returns:

The created Operation.

`restore_state`

```
restore_state(
    state,
    name=None
)
```

Restore a reader to a previously saved state.

Not all Readers support being restored, so this can produce an Unimplemented error.

Args:

- `state` : A string Tensor. Result of a SerializeState of a Reader with matching type.
- `name` : A name for the operation (optional).

Returns:

The created Operation.

`serialize_state`

```
serialize_state(name=None)
```

Produce a string tensor that encodes the state of a reader.

Not all Readers support being serialized, so this can produce an Unimplemented error.

Args:

- `name` : A name for the operation (optional).

Returns:

A string Tensor.