tf.contrib.lookup.MutableDenseHashTable

Contents

Class MutableDenseHashTable
Properties
init
key_dtype

Class `MutableDenseHashTable`

Inherits From: `LookupInterface`

Defined in `tensorflow/contrib/lookup/lookup_ops.py`.

A generic mutable hash table implementation using tensors as backing store.

Data can be inserted by calling the insert method. It does not support initialization via the init method.

It uses "open addressing" with quadratic reprobing to resolve collisions. Compared to `MutableHashTable` the insert and lookup operations in a `MutableDenseHashTable` are typically faster, but memory usage can be higher. However, `MutableDenseHashTable` does not require additional memory for temporary tensors created during checkpointing and restore operations.

Example usage:

```
table = tf.contrib.lookup.MutableDenseHashTable(key_dtype=tf.int64,
                                                value_dtype=tf.int64,
                                                default_value=-1,
                                                empty_key=0)
table.insert(keys, values)
out = table.lookup(query_keys)
print(out.eval())
```

Properties

`init`

The table initialization op.

`key_dtype`

The table key dtype.

`name`

The name of the table.

`value_dtype`

The table value dtype.

Methods

`__init__`

```
__init__(
    key_dtype,
    value_dtype,
    default_value,
    empty_key,
    initial_num_buckets=None,
    shared_name=None,
    name='MutableDenseHashTable',
    checkpoint=True
)
```

Creates an empty `MutableDenseHashTable` object.

Creates a table, the type of its keys and values are specified by key_dtype and value_dtype, respectively.

Args:

- `key_dtype`: the type of the key tensors.
- `value_dtype`: the type of the value tensors.
- `default_value`: The value to use if a key is missing in the table.
- `empty_key`: the key to use to represent empty buckets internally. Must not be used in insert or lookup operations.
- `initial_num_buckets`: the initial number of buckets.
- `shared_name`: If non-empty, this table will be shared under the given name across multiple sessions.
- `name`: A name for the operation (optional).
- `checkpoint`: if True, the contents of the table are saved to and restored from checkpoints. If `shared_name` is empty for a checkpointed table, it is shared using the table node name.

Returns:

A `MutableHashTable` object.

Raises:

- `ValueError`: If checkpoint is True and no name was specified.

`export`

```
export(name=None)
```

Returns tensors of all keys and values in the table.

Args:

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

Returns:

A pair of tensors with the first tensor containing all keys and the second tensors containing all values in the table.

`insert`

```
insert(
    keys,
    values,
    name=None
)
```

Associates **`keys`** with **`values`**.

Args:

- `keys`: Keys to insert. Can be a tensor of any shape. Must match the table's key type.
- `values`: Values to be associated with keys. Must be a tensor of the same shape as **`keys`** and match the table's value type.
- `name`: A name for the operation (optional).

Returns:

The created Operation.

Raises:

- `TypeError`: when **`keys`** or **`values`** doesn't match the table data types.

`lookup`

```
lookup(
    keys,
    name=None
)
```

Looks up **`keys`** in a table, outputs the corresponding values.

The **`default_value`** is used for keys not present in the table.

Args:

- `keys`: Keys to look up. Can be a tensor of any shape. Must match the table's key_dtype.
- `name`: A name for the operation (optional).

Returns:

A tensor containing the values in the same shape as **`keys`** using the table's value type.

Raises:

- `TypeError`: when `keys` do not match the table data types.

`size`

```
size(name=None)
```

Compute the number of elements in this table.

Args:

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

Returns:

A scalar tensor containing the number of elements in this table.
