# FrovedisSparseData

## NAME

FrovedisSparseData - A data structure used in modeling the in-memory sparse data of frovedis server side at client spark side.

## SYNOPSIS

import com.nec.frovedis.exrpc.FrovedisSparseData

### Constructors

FrovedisSparseData (`RDD[Vector]` data)

### Public Member Functions

Unit load (`RDD[Vector]` data)
Unit loadcoo (`RDD[Rating]` data)
Unit debug_print()
Unit release()

## DESCRIPTION

FrovedisSparseData is a pseudo sparse structure at client spark side which aims to model the frovedis server side sparse data (basically crs matrix).

Note that the actual sparse data is created at frovedis server side only. Spark side FrovedisSparseData contains a proxy handle of the in-memory sparse data created at frovedis server, along with number of rows and number of columns information.

### Constructor Documentation

#### FrovedisSparseData (`RDD[Vector]` data)

It accepts a spark-side RDD data of sparse or dense Vector and converts it into the frovedis server side sparse data whose proxy along with number of rows and number of columns information are stored in the constructed FrovedisSparseData object.

For example,

```
// sample input matrix file with elements in a row separated by whitespace
val data = sc.textFile(input)
// parsedData: RDD[Vector]
val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble)))
// conversion of spark data to frovedis side sparse data
val fdata = new FrovedisSparseData(parsedData)
```

## Pubic Member Function Documentation

### Unit load (RDD[Vector] data)

This function can be used to load a spark side sparse data to a frovedis server side sparse data (crs matrix). It accepts a spark-side RDD data of sparse or dense Vector and converts it into the frovedis server side sparse data whose proxy along with number of rows and number of columns information are stored in the target FrovedisSparseData object.

For example,

```
// sample input matrix file with elements in a row separated by whitespace
val data = sc.textFile(input)
// parsedData: RDD[Vector]
val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble)))

val fdata = new FrovedisSparseData() // an empty object
// conversion of spark data to frovedis side sparse data
fdata.load(parsedData)
```

### Unit loadcoo (RDD[Rating] data)

This function can be used to load a spark side Rating matrix (COO data) to a frovedis server side sparse data (crs matrix). It accepts a spark-side RDD[Rating] object and converts it into the frovedis server side sparse data whose proxy along with number of rows and number of columns information are stored in the target FrovedisSparseData object.

For example,

```
// sample input matrix file with rows of COO triplets (i,j,k)
val data = sc.textFile(input)
// ratings: RDD[Rating]
val ratings = data.map(_.split(',') match { case Array(user, item, rate) =>
                      Rating(user.toInt, item.toInt, rate.toDouble)
              })

val fdata = new FrovedisSparseData() // an empty object
// conversion of spark coo data to frovedis side sparse (crs) data
fdata.loadcoo(ratings)
```

### Unit debug_print()

It prints the contents of the server side sparse data on the server side user terminal. It is mainly useful for debugging purpose.

**Unit release()**

This function can be used to release the existing in-memory data at frovedis server side.