Package: raven.rdf (via r-universe)

October 13, 2024

Title An R Interface for Raven DataFrames (Beta0)

Version 0.2.0

Description Provides an I/O interface between R data.frames and Raven DataFrames. Defines functions to both read and write DataFrame files, as well as serialize/deserialize data.frames/DataFrames.

License Apache License (== 2)

URL https://github.com/raven-computing/rdf

Encoding UTF-8

Depends R (>= 3.5.0)

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Maintainer Phil Gaiser <phil.gaiser@raven-computing.com>

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://raven-computing.r-universe.dev

RemoteUrl https://github.com/raven-computing/rdf

RemoteRef HEAD

RemoteSha 229d452aea304d0fae5aeb9894169b5732197b39

Contents

`deserializeDataFrame` *Deserializes the given vector of raw bytes and returns a data.frame object.*

Description

The raw vector to be deserialized must represent a Raven DataFrame. That DataFrame is returned as an R data.frame object.

Usage

```
deserializeDataFrame(bytes)
```

Arguments

`bytes`	The vector of raw bytes to deserialize

Details

The column types from Raven DataFrames are mapped to the corresponding R types. More specifically, all integer types (byte, short, int, long) are mapped to the R 'integer' type. The floating point types (float, double) are mapped to the R 'double' type. Both string and char types are mapped to the R 'character' type. Booleans are mapped to the R 'logical' type. Binary columns are represented as R 'list' types containing raw vectors.

Value

A data.frame object from the specified raw vector

See Also

`readDataFrame()` for reading DataFrame (.df) files directly.

Examples

```
## Not run:
# deserialize a raw vector representing a DataFrame
df <- deserializeDataFrame(my.raw.vector)

# get the types for all columns
types <- sapply(df, typeof)

## End(Not run)
```

readDataFrame *Reads a DataFrame from the specified file.*

Description

The file to be read must be a DataFrame (.df) file. The content of the file is returned as an R data.frame object.

Usage

```
readDataFrame(filepath)
```

Arguments

filepath	The path to the file to read

Details

The column types from Raven DataFrames are mapped to the corresponding R types. More specifically, all integer types (byte, short, int, long) are mapped to the R 'integer' type. The floating point types (float, double) are mapped to the R 'double' type. Both string and char types are mapped to the R 'character' type. Booleans are mapped to the R 'logical' type. Binary columns are represented as R 'list' types containing raw vectors.

Value

A data.frame object

See Also

deserializeDataFrame() for deserializing vectors of raw bytes.
writeDataFrame() for writing DataFrame files which can be read by this function.

Examples

```
## Not run:
# read a .df file into memory
df <- readDataFrame("/path/to/my/file.df")

# get the types for all columns
types <- sapply(df, typeof)

## End(Not run)
```

`serializeDataFrame` *Serializes the specified data.frame object to a vector of raw bytes.*

Description

The R data.frame is serialized as a Raven DataFrame. The concrete column types to use for each individual data.frame column can be specified by the 'types' argument.

Usage

```
serializeDataFrame(df, types = NULL, compress = FALSE, as.nullable = FALSE)
```

Arguments

`df`	The data.frame object to serialize
`types`	The type names for all column types. Must be a vector of character values. May be NULL
`compress`	A logical indicating whether to compress the content of the returned raw vector
`as.nullable`	A logical indicating whether the data.frame should be serialized as a Nullable-DataFrame, even if it contains no NA values

Details

The column types of the R data.frame object are mapped to the corresponding Raven DataFrame column types. The following types exist:

Type name	Description
byte	int8
short	int16
int	int32
long	int64
float	float32
double	float64
string	UTF-8 encoded unicode string
char	single printable ASCII character
boolean	logical value TRUE or FALSE
binary	arbitrary length byte array

By default, if the 'types' argument is not explicitly specified, all values are mapped to the corresponding largest possible type in order to avoid possible loss of information. However, users can specify the concrete type for each column in the DataFrame file to be written. This is done by providing a vector of character values denoting the type name of each corresponding data.frame column. The index of each entry corresponds to the index of the column in the underlying data.frame to persist.

If the specified data.frame object contains at least one NA value, then the serialized DataFrame will represent a NullableDataFrame. If the data.frame contains no NA values, then the serialized DataFrame will represent a DefaultDataFrame, unless the 'as.nullable' argument is set to TRUE.

The logical 'compress' argument specifies whether the serialized DataFrame is compressed.

Value

A raw vector representing the serialized date.frame object

See Also

`writeDataFrame()` for directly persisting data.frame objects to the file system

Examples

```
## Not run:
# get a data.frame
df <- cars
# serialize the data.frame to a raw vector
vec <- serializeDataFrame(df)

# specify the concrete types of all columns
coltypes <- c("float", "double")
# serialize the data.frame to a raw vector with concrete types
serializeDataFrame(df, types = coltypes)

## End(Not run)
```

`writeDataFrame` *Writes the specified data.frame to the specified file.*

Description

The R data.frame is persisted as a DataFrame (.df) file. The concrete column types to use for each individual data.frame column can be specified by the 'types' argument.

Usage

```
writeDataFrame(filepath, df, types = NULL, as.nullable = FALSE)
```

Arguments

`filepath`	The path to the file to write
`df`	The data.frame object to write
`types`	The type names for all column types. Must be a vector of character values. May be NULL
`as.nullable`	A logical indicating whether the data.frame should be persisted as a NullableDataFrame, even if it contains no NA values

Details

The column types of the R data.frame object are mapped to the corresponding Raven DataFrame column types. The following types exist:

Type name	Description
byte	int8
short	int16
int	int32
long	int64
float	float32
double	float64
string	UTF-8 encoded unicode string
char	single printable ASCII character
boolean	logical value TRUE or FALSE
binary	arbitrary length byte array

By default, if the 'types' argument is not explicitly specified, all values are mapped to the corresponding largest possible type in order to avoid possible loss of information. However, users can specify the concrete type for each column in the DataFrame file to be written. This is done by providing a vector of character values denoting the type name of each corresponding data.frame column. The index of each entry corresponds to the index of the column in the underlying data.frame to persist.

If the specified data.frame object contains at least one NA value, then the DataFrame file to be persisted will represent a NullableDataFrame. If the data.frame contains no NA values, then the DataFrame file to be persisted will represent a DefaultDataFrame, unless the 'as.nullable' argument is set to TRUE.

Value

The number of bytes written to the specified file

See Also

`serializeDataFrame()` for serializing data.frame objects to vectors of raw bytes.
`readDataFrame()` for reading DataFrame files which have been previously persisted by this function.

Examples

```
## Not run:
# get a data.frame
df <- cars
# write the data.frame to a .df file
writeDataFrame("cars.df", df)

# specify the concrete types of all columns
coltypes <- c("float", "double")
# write the data.frame to a .df file with concrete types
```

```
writeDataFrame("cars.df", df, types = coltypes)

## End(Not run)
```

Index