Package ‘AdapteR’

February 21, 2016

Title This package wraps around Fuzzy Logix's DBLytix functions

Version 2.0

Description The DB Lytix(TM) suite of functions offers scalable and robust high performance analytical methods that are embedded seamlessly into database systems. Package RLytix makes it possible to use this functionality from R by providing wrapper functions around the SQL interface of DB Lytix(TM).

Depends R (>= 3.2.0),
plyr,

Imports MASS (>= 7.3-10),
Matrix (>= 1.1-5),
base,
stats,
psych,
reshape2,
cluster

License GPL-2

LazyData true

Collate 'FLDims.R'
'FLIs.R'
'FLPrint.R'
'FLStore.R'
'utilities.R'
'FLMatrix.R'
'FLTable.R'
'FLVector.R'
'FLCastFunctions.R'
'FLCbind.R'
'FLCholeskyDecomp.R'
'FLColMeans.R'
'FLColSums.R'
'FLCorrel.R'
'FLDet.R'
'FLDiag.R'
'FLEigen.R'
'FLGinv.R'
'FLHessenDecomp.R'
'FLIdentical.R'
'FLJordanDecomp.R'

'FLKMeans.R'
'FLLUDecomp.R'
'FLLength.R'
'data_prep.R'
'FLLinRegr.R'
'FLLogRegr.R'
'FLMatrixArithmetic.R'
'FLconstructSQL.R'
'FLMatrixBind.R'
'FLMatrixClasses.R'
'FLMatrixNorm.R'
'FLMatrixREF.R'
'FLMatrixRREF.R'
'FLQRDecomp.R'
'FLRankMatrix.R'
'FLRbind.R'
'FLRowMeans.R'
'FLRowSums.R'
'FLSV.R'
'FLSVDecomp.R'
'FLSolve.R'
'FLSolveExcl.R'
'FLSubsetting.R'
'FLTrace.R'
'FLTranspose.R'
'FLTriDiag.R'

RoxygenNote 5.0.1

R topics documented:

* *Element-Wise Multiplication of in-database objects.*

Description

* does the Element-wise Multiplication of in-database objects.

Usage

```
"*"(pObj1, pObj2)
```

Arguments

x — can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

y — can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Element-wise Multiplication of in-database objects mimics the normal Element-wise Multiplication of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

* returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 1,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
ResultFLmatrix <- flmatrix * Rvector
```

+ *Addition of in-database objects.*

Description

+ does the addition of in-database objects.

Usage

```
"+"(pObj1, pObj2)
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The addition of in-database objects mimics the normal addition of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

+ returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 1,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
ResultFLmatrix <- flmatrix + Rvector
```

/ *Element-wise Division of in-database objects.*

Description

/ does the Element-wise Division of in-database objects.

Usage

```
"/"(pObj1, pObj2)
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Element-wise Division of in-database objects mimics the / of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

/ returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Constraints

division by 0 gives inf in R, but is not supported for in-database objects

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 1,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
ResultFLmatrix <- flmatrix / Rvector
```

`==.FLMatrix` *Equality of in-database objects.*

Description

== checks the equality of in-database objects.

Usage

```
## S3 method for class 'FLMatrix'
pObj1 == pObj2
```

Arguments

pObj1	can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
pObj2	can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The equality of in-database objects mimics the normal addition of R data types. One can check equality of FLMatrices, FLMatrix - R matrices, FLVectors and FLVector - RVector.

Value

== returns a logical TRUE or FALSE.

Constraints

Currently only dgCMatrix,dgeMatrix,dsCMatrix, dgTMatrix,matrix,Matrix,vector R types are supported. Comparision of FLMatrix with FLVector is not currently Supported. In case of FLVector and Rvector comparision use FLVector==RVector in place of RVector==FLVector

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
flvector <- as.FLVector(1:5,connection)
Result <- flmatrix == flmatrix
Result <- flvector==flvector
Result <- flvector==1:5
```

as.data.frame *Converts in-database objects to a data frame in R*

Description

Caution: data is fetched into R session

Usage

```
as.data.frame(x, ...)
```

Arguments

x	can be FLTable, FLVector or FLMatrix

as.FLMatrix *Casting to FLMatrix*

Description

Converts input to a FLMatrix object

Usage

```
as.FLMatrix(object, connection, sparse = TRUE, ...)
```

Arguments

`object` matrix, vector, data frame, sparseMatrix, FLVector which needs to be casted to FLMatrix and inserted in-database

`connection` ODBC/JDBC connection object

`sparse` logical if sparse representation to be used

`...`

`nr` number of rows in resulting FLMatrix

`nc` number of columns in resulting FLMatrix. nr and nc inputs are applicable only in case of vector,FLVector

Value

FLMatrix object after casting.

`as.FLVector` *casting to FLVector*

Description

Converts input `obj` to FLVector object

Usage

```
as.FLVector(object, connection, ...)
```

Arguments

`object` matrix, vector, data frame, sparseMatrix, FLMatrix which needs to be casted to FLVector

`connection` ODBC/JDBC connection object

`...` additional arguments like size

`size` number of elements in resulting FLVector. size input is not applicable only in case of FLMatrix

Value

FLVector object after casting.

`as.matrix` *Converts in-database objects to a matrix in R*

Description

Caution: data is fetched into R session

Usage

```
as.matrix(x, ...)
```

Arguments

`x`	can be FLTable, FLVector or FLMatrix

`as.matrix.FLVector` *Converts FLVector object to a matrix in R*

Description

Converts FLVector object to a matrix in R

Usage

```
## S3 method for class 'FLVector'
as.matrix(obj)
```

`as.vector.FLMatrix` *Converts FLMatrix object to vector in R*

Description

Converts FLMatrix object to vector in R

Usage

```
as.vector.FLMatrix(object, mode = "any")
```

`as.vector.FLVector` *Converts FLVector object to vector in R*

Description

Converts FLVector object to vector in R

Usage

```
as.vector.FLVector(object, mode = "any")
```

cbind *Combine objects by columns.*

Description

cbind combines input objects by columns and forms a FLMatrix.

Usage

```
cbind(x, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

cbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by columns and makes a FLMatrix.

Value

cbind returns a FLMatrix object which is the column-wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of rows.

cbind.FLMatrix *Combine objects by columns.*

Description

cbind combines input objects by columns and forms a FLMatrix.

Usage

```
## S3 method for class 'FLMatrix'
cbind(object, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

cbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by columns and makes a FLMatrix.

Value

`cbind` returns a FLMatrix object which is the column-wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of rows.

`checkSameDims` *Compare Matrix Dimensions*

Description

Takes two matrices in-database or R, and returns true if they have same dimensions

Usage

```
checkSameDims(object1, object2)
```

Arguments

`object1`	FLMatrix or R Matrix
`object2`	FLMatrix or R Matrix

Value

logical

`chol` *Cholesky Decomposition.*

Description

`chol` computes the Cholesky factorization of FLMatrix object.
The Cholesky decomposition is a decomposition of a positive definite matrix into the product of a lower triangular matrix and its conjugate transpose.

Usage

```
chol(x, ...)
```

Arguments

`x`	is of class FLMatrix

Details

The wrapper overloads chol and implicitly calls FLCholeskyDecompUdt.

Value

chol returns FLMatrix which is the upper triangular factor of the Cholesky decomposition

Constraints

Input can only be a Hermitian, positive definite square matrix (n x n) with maximum dimension limitations of (1000 x 1000)

Examples

```
connection<-odbcConnect("Gandalf")
flmatrix<-FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- chol(flmatrix)
```

---

colMeans *column means of a FLMatrix.*

Description

colMeans computes the column-wise average of FLMatrix objects.

Usage

```
colMeans(x, ...)
```

Arguments

x	is of class FLMatrix.

Value

colMeans returns a FLVector object representing the column-wise Means.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- colMeans(flmatrix)
```

colSums *column sums of a FLMatrix.*

Description

`colSums` computes the column-wise sums of FLMatrix objects.

Usage

```
colSums(x, ...)
```

Arguments

`x`	is of class FLMatrix.

Value

`colSums` returns a FLVector object representing the col-wise sums.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- colSums(flmatrix)
```

constructSelect *constructs a sql statement returning the deep table representation of the object.*

Description

constructs a sql statement returning the deep table representation of the object.

Usage

```
constructSelect(object, ...)
```

Arguments

`object`	the object to query
`...`	arguments passed on to SQL generation. see joinNames

Value

a character SQL representation

cor *Correlation.*

Description

cor computes correlation of in-database Objects

Usage

```
cor(x, y, ...)
```

Arguments

x	FLMatrix, FLVector or FLTable object or any R object
y	FLMatrix, FLVector or FLTable object or any R object

Value

cor returns FLMatrix object representing correlation of x and y.

Constraints

The number of non-null pairs must be greater than or equal to 2. If number of non-null pairs is less than 2, FLCorrel returns a NULL.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
deeptable <- FLTable(connection, "FL_DEMO", "tblUSArrests", "ObsID","VarID","Num_Val")
widetable <- FLTable(connection,"FL_DEMO","tblAbaloneWide","ObsID")
cor(deeptable,deeptable)
cor(widetable,widetable)
```

deepToWide *Convert Deep Table to Wide Table*

Description

Convert Deep Table to Wide Table

Usage

```
deepToWide(object, whereconditions, mapTable, mapName, outWideTableDatabase,
  outWideTableName)
```

Arguments

`object` FLTable object to convert to wide table

`whereconditions` character vector specifying whereconditions if any to reference the input deep table

`mapTable` name of the in-database table containing mapping information to be used in conversion if any

`mapName` unique identifier for the mapping information in mapping table if any

`outWideTableDatabase` name of database where output wide table is to be stored

`outWideTableName` name to give to the output wide table

Value

`deepToWide` returns a list containing components `table` which is the FLTable referencing the wide table and `AnalysisID` giving the AnalysisID of conversion

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
deeptable  <- FLTable(connection, "FL_DEMO", "tblUSArrests", "ObsID","VarID","Num_Val")
resultList <- deepToWide(deeptable)
widetable <- resultList$table
analysisID <- resultList$AnalysisID
```

`det` *Determinant of a Matrix.*

Description

`det` computes the determinant of FLMatrix objects.

Usage

```
det(x, ...)
```

Arguments

`x` is a FLMatrix object

Value

`det` returns determinant as a R vector which replicates the equivalent R vector output.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLvector <- det(flmatrix)
```

---

diag *Matrix Diagonals*

Description

Extract or replace the diagonal of a matrix, or construct a diagonal matrix.

Usage

```
diag(x, ...)
```

Arguments

x	is an object of class FLMatrix or FLVector

Details

diag has three distinct usages: x is a FLMatrix, when it extracts the diagonal. x is a scalar (length-one FLVector) and the only argument, it returns a square identity matrix of size given by the scalar. x is a FLVector, either of length at least 2. This returns a square matrix with the given diagonal entries.

Value

If x is a FLMatrix then diag(x) returns the diagonal of x as FLVector object. If x is FLVector, the value is a diagonal square FLMatrix with diagonal elements as given in FLVector.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- diag(flmatrix)
DeepTable <- FLTable(connection, "FL_DEMO", "tblUSArrests","ObsID")
flvectorDeep <- DeepTable[1:5,1]
resultFLMatrix <- diag(flvectorDeep)
```

drop.FLTable *drop a table*

Description

drop a table

Usage

```
drop.FLTable(object)
```

Arguments

object	FLTable object

Value

message if the table is dropped

eigen *Spectral Decomposition of a Matrix.*

Description

eigen Computes eigenvalues and eigenvectors of FLMatrices.

Usage

```
eigen(x, ...)
```

Arguments

x	is of class FLMatrix

Value

eigen returns a list of FLMatrix object containing the eigen vectors and a FLVector object containing eigen values which replicates the equivalent R output.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000). Complex Eigen values and vectors are not Supported.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultList <- eigen(flmatrix)
resultList$values
resultList$vectors
```

FLamendDimnames *Amends dimension names to a matrix if a mapping exists.*

Description

If no mapping exists uses the unique row and column names. Dimnames 1:n are set to NULL in order to adhere to R conventions.

Usage

```
FLamendDimnames(flm, map_table)
```

Arguments

flm	FLMatrix

Value

the FLMatrix object, with slot dimnames re set

FLHessen *Hessenberg Decomposition of a Matrix.*

Description

`FLHessen` computes the Hessenberg decomposition for FLMatrix objects.

Usage

```
FLHessen(x, ...)
```

Arguments

x	is of class FLMatrix

Value

`FLHessen` returns a list of two components:

P	FLMatrix representing P matrix obtained from Hessenberg decomposition
H	FLMatrix representing H matrix obtained from Hessenberg decomposition

Constraints

Input can only be square matrix with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultList <- FLHessen(flmatrix)
resultList$P
resultList$H
```

`FLJordan` *Jordan Decomposition of a Matrix.*

Description

`FLJordan` computes the Jordan decomposition for FLMatrix objects.

Usage

```
FLJordan(x, ...)
```

Arguments

`x`	is of class FLMatrix

Value

`FLJordan` returns a list of two components:

`J`	FLVector representing J vector obtained from Jordan decomposition
`P`	FLMatrix representing P matrix obtained from Jordan decomposition
`PInv`	FLMatrix representing PInv matrix obtained from Jordan decomposition

Constraints

Input can only be square matrix with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultList <- FLJordan(flmatrix)
resultList$J
resultList$P
resultList$PInv
```

`FLKMeans-class` *An S4 class to represent FLKMeans*

Description

An S4 class to represent FLKMeans

Arguments

`object`	retrieves the cluster vector
`object`	retrieves the coordinates of the centroids
`object`	overloads the print function
`object`	total within sum of squares
`object`	within sum of squares
`object`	between sum of squares
`object`	total sum of squares
`object`	size vector

Slots

`no_of_centers` A numeric vector containing the number of clusters, say k

`AnalysisID` A character output used to retrieve the results of analysis

`connection` ODBC connectivity for R

`table` FLTable object given as input on which analysis is performed

`resultsfetched` A logical vector describing what components are fetched

`results` A list of all fetched components

`deeptablename` A character vector containing a deeptable(either conversion from a widetable or input deeptable)

`FLLinRegr-class` *An S4 class to represent FLLinRegr*

Description

An S4 class to represent FLLinRegr

Arguments

`object`	contains: call,coefficients
`object`	a named vector of coefficients
`object`	contains: call,residuals,coefficients,significant codes note and statistical output.

Slots

`formula` an object of class 'formula': Model Formulae

`table_name` A character

`deeptablename` A character vector containing name of the deeptable on conversion from a widetable

`AnalysisID` An output character ID from CALL FLLinRegr

`dataprepID` An output character ID from CALL FLRegrDataPrep

`datatable` An object of class FLTable

`FLLogRegr-class` *An S4 class to represent FLLogRegr*

Description

An S4 class to represent FLLogRegr

Arguments

`object`	contains: call,coefficients
`object`	a named vector of coefficients
`object`	contains: call,residuals,coefficients,significant codes note and statistical output.

Slots

`formula` an object of class 'formula': Model Formulae

`table_name` A character

`deeptablename` A character vector containing name of the deeptable on conversion from a widetable

`AnalysisID` An output character ID from CALL FLLogRegr

`dataprepID` An output character ID from CALL FLRegrDataPrep

`datatable` An object of class FLTable

`FLLU-class` *An S4 class to represent LU Decomposition*

Description

An S4 class to represent LU Decomposition

Slots

`x` object of class FLVector

`perm` object of class FLVector

`Dim` object of class FLVector

`lower` object of class FLMatrix

`upper` object of class FLMatrix

`data_perm` object of class FLMatrix

FLMatrix *Constructor function for FLMatrix.*

Description

FLMatrix constructs an object of class FLMatrix.

Usage

```
FLMatrix(connection, database = getOption("ResultDatabaseFL"), table_name,
  matrix_id_value = "", matrix_id_colname = "",
  row_id_colname = "rowIdColumn", col_id_colname = "colIdColumn",
  cell_val_colname = "valueColumn", dim = 0, dimnames = NULL,
  conditionDims = c(FALSE, FALSE), whereconditions = c(""),
  map_table = NULL)
```

Arguments

connection ODBC/JDBC connection handle to the database.

database name of the database

table_name name of the matrix table

matrix_id_value identifier for the input matrix

matrix_id_colname matrix id value in table_name

row_id_colname column name in table_name where row numbers are stored

col_id_colname column name in table_name where column numbers are stored

cell_val_colname column name in table_name where matrix elements are stored

dim vector representing the dimensions of the matrix

dimnames list of dimension names to assign to the matrix

conditionDims logical vector of length two representing if there are any conditions on dimensions

whereconditions where conditions if any to reference the in-database matrix

map_table in-database table name which stores the dimnames of the matrix. Not required if dimnames are already specified using dimnames

Details

FLMatrix object is an in-database equivalent to matrix object. This object is used as input for matrix operation functions.

Value

FLMatrix returns an object of class FLMatrix mapped to an in-database matrix.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5, "Matrix_id","ROW_ID","COL_ID","CELL_VAL")
flmatrix
```

`FLMatrix-class` *An S4 class to represent FLMatrix. A Matrix can be either based off a query from a deep table (customizable by any where-condition) – or based off an arbitrary SQL statement returning a deep table.*

Description

An S4 class to represent FLMatrix. A Matrix can be either based off a query from a deep table (customizable by any where-condition) – or based off an arbitrary SQL statement returning a deep table.

Slots

`dimnames` list of 2 elements with row, column names of FLMatrix object

`dim` list of 2 FLTableQuery instances (or NULL) that map row_ids in the select to row-names in R

`FLMatrixBind` *Bind a matrix/array by an index. Currently limited to matrices with character dimnames*

Description

Bind a matrix/array by an index. Currently limited to matrices with character dimnames

Usage

```
FLMatrixBind(parts, by)
```

Arguments

`parts`	
`by`	the numeric index by which binding takes place

Value

returns a remote matrix object defining the deep table sql for the *bound result.

FLMatrixNorm *Norm of a Matrix.*

Description

`FLMatrixNorm` gives the value of Norm for FLMatrix objects.

Usage

```
FLMatrixNorm(x, ...)
```

Arguments

`x`	is of class FLMatrix
`NormMethod`	is an integer from 1-4 representing the type of norm that should be computed.

Value

`FLMatrixNorm` returns a R vector object which is the Norm of input FLMatrix object calculated using method specified by NormMethod input. There are 4 types of norms of a matrix:

`1-Norm`	Maximum of the sum of the absolute values for the columns
`2-Norm`	Maximum of the sum of the absolute values for the rows
`Frobenius Norm`	Square root of the trace of (t(A)A)
`Infinity Norm`	Square root of the maximum of the magnitudes of the Eigenvalues of (t(A)A)

Constraints

Input can only be with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- FLMatrixNorm(flmatrix,4)
```

FLMatrixREF *Row Echelon form of a Matrix.*

Description

`FLMatrixREF` gives the Row Echelon form of FLMatrix objects.

Usage

```
FLMatrixREF(x, ...)
```

Arguments

x	is of class FLMatrix

Value

FLMatrixREF returns a FLMatrix object which is the Row Echelon form of input FLMatrix.

Constraints

Input can only be a square FLMatrix with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- FLMatrixREF(flmatrix)
```

FLMatrixRREF *Reduced Row Echelon form of a Matrix.*

Description

FLMatrixRREF gives the Reduced Row Echelon form of FLMatrix objects.

Usage

```
FLMatrixRREF(x, ...)
```

Arguments

x	is of class FLMatrix

Value

FLMatrixRREF returns a FLMatrix object which is the Reduced Row Echelon form of input FLMatrix.

Constraints

Input can only be a square FLMatrix with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- FLMatrixRREF(flmatrix)
```

FLodbcClose — *Close Session and Drop temp Tables*

Description

Strongly recommended to run before quitting current R session

Usage

```
FLodbcClose(connection)
```

Arguments

connection	ODBC/JDBC connection object

FLSelectFrom-class — *A selectFrom models a select from a table*

Description

A selectFrom models a select from a table

Slots

`database` character

`table_name` character

FLSolveExcl — *Inverse of a Matrix excluding a dimension.*

Description

`FLSolveExcl` computes the inverse for FLMatrix objects by excluding the specified row and column from the matrix.

Usage

```
FLSolveExcl(x, ExclIdx, ...)
```

Arguments

x	is of class FLMatrix
ExclIdx	is a positive integer specifying row or column id to be excluded.

Value

`solveExcl` returns a FLMatrix object which is the inverse of input FLMatrix object after excluding given dimension.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- solveExcl(flmatrix,3)
```

FLStartSession *Starts Session and Creates temp Tables for result storage*

Description

Strongly recommended to run before beginning a new R session use options to specify the following:- ResultDatabaseFL, ResultVectorTableFL, ResultMatrixTableFL, MatrixNameMapTableFL, ResultSparseMatrixTableFL

Usage

```
FLStartSession(connection, database = "FL_DEMO", persistent = "test",
  drop = TRUE,
  tableoptions = ", FALLBACK ,\n\t\t\t\t    NO BEFORE JOURNAL,\n\t\t\t\t    NO AFTER JOURNAL,\n\t\
```

Arguments

`connection`	ODBC/JDBC connection object
`database`	name of current database
`persistent`	NULL if result tables are to be created as volatile tables
`drop`	logical to specify to drop result tables if already existing
`tableoptions`	options used to create result tables

FLSV *Singular Values of a FLMatrix.*

Description

`FLSV` computes the singular values for FLMatrix objects.

Usage

```
FLSV(x, ...)
```

Arguments

`object`	is of class FLMatrix

Value

FLSV returns a FLVector object representing the singular values.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- FLSV(flmatrix)
```

FLTable *Constructor function for FLTable.*

Description

FLTable constructs an object of class FLTable.

Usage

```
FLTable(connection, database, table, obs_id_colname,
  var_id_colnames = character(0), cell_val_colname = character(0),
  whereconditions = character(0))
```

Arguments

connection	ODBC/JDBC connection object
database	name of the database
table	name of the table
obs_id_colname	column name set as primary key
cell_val_colname	column name where cell values are stored if FLTable is deep
whereconditions	whereconditions if any to reference the table
var_id_colname	column name where variable id's are stored if FLTable is deep

Details

FLTable refers to an in-database table. This is equivalent to data.frame object. This object is commonly used as input for data mining functions.

Value

FLTable returns an object of class FLTable mapped to a table in Teradata.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_Deep", "tblAbaloneWide", "ObsID")
deeptable <- FLTable(connection,"FL_DEMO","tblUSArrests","ObsID","VarID","Num_Val")
names(widetable)
```

`FLTable-class` *An S4 class to represent FLTable*

Description

An S4 class to represent FLTable

Arguments

`object`	retrieves the column names of FLTable object

Slots

`db_name` character

`table_name` character

`obs_id_colname` character

`var_id_colnames` character

`cell_val_colname` character

`isDeep` logical

`FLTableFunctionQuery-class` *A TableFunctionQuery models a select from an arbitrary query*

Description

A TableFunctionQuery models a select from an arbitrary query

Slots

`SQLquery` character

FLTriDiag *TriDiagonal or Upper Hessenberg matrix of a FLMatrix.*

Description

`FLTriDiag` computes the TriDiagonal or Upper Hessenberg matrix of FLMatrix object.

Usage

```
FLTriDiag(x, ...)
```

Arguments

`x`	is of class FLMatrix

Value

`FLTriDiag` returns a FLMatrix object representing the upper Hessenberg or TriDiagonal matrix.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- FLTriDiag(flmatrix)
```

FLVector *Constructor function for FLVector, representing a vector in database*

Description

Please use subsetting of FLTable to create FLVector object

Usage

```
FLVector(table, val_col_name = character(), val_row_name = character(),
  whereconditions = character())
```

Value

`FLVector` returns an object of class FLVector mapped to an in-database vector.

See Also

`FLTable`

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
WideTable <- FLTable(connection, "FL_DEMO", "tblAbaloneWide","ObsID")
flvectorColumn <- FLTable[,"Diameter"]
flvectorRow <- FLTable[3,]
flvectorRow
flvectorColumn
```

FLVector-class *An S4 class to represent FLVector*

Description

An S4 class to represent FLVector

getMaxMatrixId *Get Max Matrix ID+1 from result Matrix table*

Description

used to know ID of next entry in table

Usage

```
getMaxMatrixId(vconnection, ...)
```

Arguments

vconnection	ODBC/JDBC connection object

getMaxValue *Get Max ID from given table*

Description

used to know ID of last entry in table

Usage

```
getMaxValue(vdatabase = getOption("ResultDatabaseFL"),
  vtable = getOption("ResultVectorTableFL"), vcolName = "vectorIdColumn",
  vconnection = vconnection)
```

Arguments

vdatabase	name of the database of table
vtable	name of the table
vcolName	name of the primary index column in table
vconnection	ODBC/JDBC connection object

getMaxVectorId *Get Max Vector ID+1 from result Vector table*

Description

used to know ID of next entry in table

Usage

```
getMaxVectorId(vconnection, ...)
```

Arguments

`vconnection`	ODBC/JDBC connection object

ginv *Generalized Inverse of a Matrix.*

Description

`ginv` computes the pseudo-inverse for FLMatrix objects.

Usage

```
ginv(x, ...)
```

Arguments

`x`	is of class FLMatrix

Value

`ginv` returns a FLMatrix object which is the pseudo-inverse of input FLMatrix object and replicates the equivalent R output.

Constraints

Input can only be with maximum dimension limitations of (500 x 500).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 1,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- ginv(flmatrix)
```

glm.FLTable *Linear Regression.*

Description

`glm` performs linear regression on FLTable objects.

Usage

```
## S3 method for class 'FLTable'
glm(formula, data, rushil = "binomial", iter = 25,
  threshold = 0.1, ...)
```

Arguments

`formula`	A symbolic description of model to be fitted
`data`	An object of class FLTable

Details

The wrapper overloads glm and implicitly calls FLRegrDataPrep and FLLogRegr.

Value

`glm` performs linear regression and replicates equivalent R output.

Constraints

None

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_REV4546", "tblAbaloneWide", "ObsID")
glmfit <- glm(Rings~Height+Diameter,widetable)
```

identical *Equality of in-database objects.*

Description

`identical` checks the equality of in-database objects.

Usage

```
identical(x, y)
```

Arguments

x can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

y can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The equality of in-database objects mimics the normal addition of R data types. One can check equality of FLMatrices, FLMatrix - R matrices, FLVectors and FLVector - RVector.

Value

`identical` returns a logical TRUE or FALSE.

Constraints

Currently only `dgCMatrix`,`dgeMatrix`,`dsCMatrix`, `dgTMatrix`,`matrix`,`Matrix`,`vector` R types are supported.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
Result <- identical(flmatrix,flmatrix)
Result <- identical(Rvector,as.FLVector(Rvector,connection))
```

`is.FLMatrix` *Check if the object is an FLMatrix object*

Description

Check if the object is an FLMatrix object

Usage

```
is.FLMatrix(object)
```

`is.FLTable` *Check if the object is an FLTable object*

Description

Check if the object is an FLTable object

Usage

```
is.FLTable(object)
```

is.FLVector *Check if the object is an FLVector object*

Description

Check if the object is an FLVector object

Usage

```
is.FLVector(object)
```

kmeans.FLTable *K-Means Clustering.*

Description

kmeans performs k-means clustering on FLTable objects.

Usage

```
## S3 method for class 'FLTable'
kmeans(table, centers, max.iter = 10, nstart = 1,
  exclude = as.character(c()), class_spec = list(), where_clause = "")
```

Arguments

table	an object of class FLTable
centers	the number of clusters
max.iter	the maximum number of iterations allowed
nstart	the initial number of random sets
exclude	the comma separated character string of columns to be excluded
class_spec	list describing the categorical dummy variables
where_clause	takes the where_clause as a string

Details

The wrapper overloads kmeans and implicitly calls FLKMeans.

Value

kmeans performs k-means clustering and replicates equivalent R output.

Constraints

None

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_TRAIN", "tblAbaloneWide", "ObsID")
kmeansobject <- kmeans(widetable,3,20,2,"Rings,SEX",list("DummyCat(D)","SEX(M)"))
print(kmeansobject)
```

length.FLMatrix *computes the length of FLMatrix object.*

Description

computes the length of FLMatrix object.

Usage

```
## S3 method for class 'FLMatrix'
length(obj)
```

Arguments

obj	is a FLMatrix object.

Value

`length` returns a R Vector giving the length of input object.

length.FLTable *computes the length of FLTable object.*

Description

computes the length of FLTable object.

Usage

```
## S3 method for class 'FLTable'
length(obj)
```

Arguments

obj	is a FLTable object.

Value

`length` returns a R Vector giving the number of observations or rows in input FLTable object.

`length.FLVector` *computes the length of FLVector object.*

Description

computes the length of FLVector object.

Usage

```
## S3 method for class 'FLVector'
length(obj)
```

Arguments

`obj`	is a FLVector object.

Value

`length` returns a R Vector giving the length of input object.

`lm.FLTable` *Linear Regression.*

Description

`lm` performs linear regression on FLTable objects.

Usage

```
## S3 method for class 'FLTable'
lm(formula, data, ...)
```

Arguments

`formula`	A symbolic description of model to be fitted
`data`	An object of class FLTable

Details

The wrapper overloads lm and implicitly calls FLRegrDataPrep and FLLinRegr.

Value

`lm` performs linear regression and replicates equivalent R output.

Constraints

None

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_REV4546", "tblAbaloneWide", "ObsID")
lmfit <- lm(Rings~Height+Diameter,widetable)
```

lu *LU Decomposition.*

Description

The LU decomposition involves factorizing a matrix as the product of a lower triangular matrix L and an upper triangular matrix U. Permutation matrix is also provided in the output. If permutation matrix is not used in the decomposition, the output of permutation matrix is an identity matrix.

Usage

```
## S3 method for class 'FLMatrix'
lu(x, ...)
```

Arguments

`x`	is of class FLMatrix

Details

`lu` replicates the equivalent lu() generic function.
`expand` decomposes the compact form to a list of matrix factors.
The expand method returns L,U and P factors as a list of FLMatrices.

The decomposition is of the form A = P L U where typically all matrices are of size (n x n), and the matrix P is a permutation matrix, L is lower triangular and U is upper triangular.

Value

`x`	the FLVector form of "L" (unit lower triangular) and "U" (upper triangular) factors of the original matrix
`perm`	FLVector that describes the permutation applied to the rows of the original matrix
`Dim`	FLVector that gives the dimension of the original matrix
`lower`	FLMatrix representing the lower triangular matrix
`upper`	FLMatrix representing the upper triangular matrix
`data_perm`	FLMatrix representing the permutation matrix

Constraints

Input can only be with maximum dimension limitations of (1000 x 1000).

Examples

```
connection<-odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
FLLUobject <- lu(flmatrix)
listresult <- expand(FLLUobject)
listresult$L
listresult$U
listresult$P
```

names.FLTable *Gives the column names of FLTable object*

Description

Gives the column names of FLTable object

Usage

```
## S3 method for class 'FLTable'
names(object)
```

Arguments

object

qr *QR Decomposition.*

Description

The QR decomposition involves factorizing a matrix into QMatrix and RMatrix.

Usage

```
qr(x, ...)
```

Arguments

x	is of class FLMatrix

Details

qr replicates the equivalent qr() generic function.

Value

qr returns a list of five components:

qr	a FLMatrix with the same dimensions as object. The upper triangle contains the R of the decomposition and the lower triangle contains information on the Q of the decomposition (stored in compact form)
qraux	a FLVector of length ncol(object) which contains additional information on Q.
rank	the FLVector giving rank of object
QMatrix	the resulting Q Matrix stored in-database as FLMatrix
RMatrix	the resulting R Matrix stored in-database as FLMatrix

Constraints

Input can only be with maximum dimension limitations of (700 x 700).

Examples

```
connection<-odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultList <- qr(flmatrix)
resultList$qr
resultList$qraux
resultList$rank
resultList$pivot
```

---

rankMatrix *Matrix Rank.*

Description

rankMatrix computes the rank of FLMatrix objects.

Usage

```
rankMatrix(x, ...)
```

Arguments

x	is of class FLMatrix

Value

rankMatrix returns R vector object of size 1 which replicates the equivalent R output.

Constraints

Input can have maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- rankMatrix(flmatrix)
```

rbind *Combine objects by rows.*

Description

rbind combines input objects by rows and forms a FLMatrix.

Usage

```
rbind(x, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

rbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by rows and makes a FLMatrix.

Value

rbind returns a FLMatrix object which is the row-wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of columns.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- rbind(flmatrix,1:5,flmatrix)
```

rbind.FLMatrix *Combine objects by rows.*

Description

rbind combines input objects by rows and forms a FLMatrix.

Usage

```
## S3 method for class 'FLMatrix'
rbind(object, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

rbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by rows and makes a FLMatrix.

Value

rbind returns a FLMatrix object which is the row wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of columns.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- rbind(flmatrix,1:5,flmatrix)
```

---

remoteTable *reference in-database object*

Description

reference in-database object

Usage

```
remoteTable(object, table)
```

Arguments

object	in-database object
table	table name. Applicable only if object is the database name.

Value

character vector giving reference to in-database object

restrictFLMatrix *Appends where clauses for subsetting etc.*

Description

Appends where clauses for subsetting etc.

Usage

```
restrictFLMatrix(object, whereconditions = object@select@whereconditions,
  dimnames = object@dimnames, conditionDims = c(FALSE, FALSE))
```

Arguments

`object`	An FLMatrix object
`whereconditions`	constraints to be added
`dimnames`	new dimension names
`conditionDims`	vector of 2 LOGICAL values, if first is TRUE, a inCondition for the rownames is appended, if 2 for the columns respectively.

rowMeans *row means of a FLMatrix.*

Description

`rowMeans` computes the row-wise average of FLMatrix objects.

Usage

```
rowMeans(x, ...)
```

Arguments

`x`	is of class FLMatrix.

Value

`rowMeans` returns a FLVector object representing the row-wise Means.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- rowMeans(flmatrix)
```

rowSums *row sums of a FLMatrix.*

Description

rowSums computes the row-wise sums of FLMatrix objects.

Usage

```
rowSums(x, ...)
```

Arguments

x	is of class FLMatrix.

Value

rowSums returns a FLVector object representing the row-wise sums.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- rowSums(flmatrix)
```

solve.FLMatrix *Inverse of a Matrix.*

Description

solve computes the inverse for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
solve(pObj1)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads solve and implicitly calls FLMatrixInvUdt.

Value

solve returns a FLMatrix object which is the inverse of input FLMatrix object and replicates the equivalent R output.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 2,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- solve(flmatrix)
```

`sqlQuery` *Send a query to database*

Description

Result is returned as data.frame

Usage

```
sqlQuery(connection, query)
```

Arguments

`query`	SQLQuery to be sent
`channel`	ODBC/JDBC connection object

`sqlSendUpdate.JDBCConnection` *Send a query to database*

Description

No result is returned

Usage

```
## S3 method for class 'JDBCConnection'
sqlSendUpdate(channel, query)
```

Arguments

`channel`	JDBC connection object
`query`	SQLQuery to be sent

`sqlSendUpdate.RODBC` *Send a query to database*

Description

No result is returned

Usage

```
## S3 method for class 'RODBC'
sqlSendUpdate(connection, query)
```

Arguments

`query`	SQLQuery to be sent
`channel`	ODBC connection object

`store` *stores an object in database*

Description

stores an object in database

Usage

```
store(object, returnType, connection, ...)
```

Arguments

`object`	the object to store. Can be FLMatrix, FLVector, FLTable, character
`returnType`	return type of the stored data. Applicable only when object is a character representing a SQL Query
`connection`	ODBC/JDBC connection object. Applicable only when object is a character representing a SQL Query

Value

in-database object after storing

svd *Singular Value Decomposition of a Matrix.*

Description

svd computes the singular value decomposition for FLMatrix objects.

Usage

```
svd(x, ...)
```

Arguments

x	is of class FLMatrix
nu	number of left singular vectors to be computed.This must between 0 and nrow(object).
nv	number of right singular vectors to be computed.This must between 0 and ncol(object).

Value

svd returns a list of three components:

d	a FLVector containing the singular values of x, of size min(n, p).
u	a FLMatrix whose columns contain the left singular vectors of x, present if nu > 0. Dimension c(n, nu).
v	a FLMatrix whose columns contain the right singular vectors of x, present if nv > 0. Dimension c(p, nv).

Constraints

Input can only be with maximum dimension limitations of (550 x 550).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultList <- svd(flmatrix)
resultList$d
resultList$u
resultList$v
```

t *Matrix Transpose.*

Description

`t` returns the transpose of FLMatrix objects.

Usage

```
t(x, ...)
```

Arguments

x	is of class FLMatrix

Value

`t` returns a FLMatrix object which is the transpose of input FLMatrix object and replicates the equivalent R output.

Constraints

Input can be a matrix of dimensions (m x n) where m > n, m < n or m = n.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLMatrix <- t(flmatrix)
```

tr *Matrix Trace.*

Description

`tr` computes the trace of FLMatrix objects.

Usage

```
tr(x, ...)
```

Arguments

x	an object of class FLMatrix

Details

`tr` computes the trace of input FLMatrix object, stores the result in-database and returns R vector object

Value

`tr` returns R Vector object of size 1 which replicates the equivalent R output.

Constraints

Input can only be with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLVector <- tr(flmatrix)
```

`viewSelectMatrix,FLMatrix,character,character-method` *todo: alias*

Description

todo: alias

Usage

```
## S4 method for signature 'FLMatrix,character,character'
viewSelectMatrix(object, localName,
  withName = "z")
```

`wideToDeep` *Convert Wide Table to Deep Table in database.*

Description

Convert Wide Table to Deep Table in database.

Usage

```
wideToDeep(object, excludeCols, classSpec, whereconditions, outDeepTableName,
  outDeepTableDatabase, outObsIDCol, outVarIDCol, outValueCol)
```

Arguments

`object`	FLTable object
`excludeCols`	character vector specifying columns to be excluded from conversion
`classSpec`	list representing Class specification which identifies then value of the categorical variable to be used a reference
`whereconditions`	character vector giving where conditions if any to reference the wide table

`outDeepTableName`
name to be given to the output deep table

`outDeepTableDatabase`
name of database to store the output deep table

`outObsIDCol` name to give to the primary key column name of the output deep table

`outVarIDCol` name to give to the varibales name column of the output deep table

`outValueCol` name to give to the value column of the output deep table

Value

`wideToDeep` returns a list containing components `table` which is the FLTable referencing the deep table and `AnalysisID` giving the AnalysisID of conversion

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_DEMO", "tblAbaloneWide", "ObsID")
resultList <- wideToDeep(widetable)
deeptable <- resultList$table
analysisID <- resultList$AnalysisID
```

`[.FLMatrix` *Extract part of FLMatrix object.*

Description

`[]` acts on FLMatrix objects and extracts parts of them.

Usage

```
## S3 method for class 'FLMatrix'
object[rows = 1, cols = 1, drop = TRUE]
```

Arguments

`object` is a FLMatrix object

`rows` is a vector input corresponding to rows to be extracted

`cols` is a vector input corresponding to columns to be extracted

Value

`[]` returns FLMatrix object after extraction which replicates the equivalent R extraction.

Constraints

Applying UDT functions on subsetted matrices with discontinuous row and col ids' may result in error

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 2,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
resultFLmatrix <- flmatrix[1,]
```

[.FLTable *Extract part of FLTable object.*

Description

`[]` acts on FLMatrix objects and extracts parts of them.

Usage

```
## S3 method for class 'FLTable'
object[rows = 1, cols = 1, drop = TRUE]
```

Arguments

`object`	is a FLTable object
`rows`	is a vector input corresponding to rows to be extracted
`cols`	is a vector input corresponding to columns to be extracted

Value

`[]` returns FLMatrix object after extraction which replicates the equivalent R extraction.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
fltable <- FLTable(connection, "FL_DEMO", "tblAbaloneWide", "ObsID")
resultFLtable <- fltable[1:10,4:6]
```

[.FLVector *Extract part of FLVector object.*

Description

`[]` acts on FLVector objects and extracts parts of them.

Usage

```
## S3 method for class 'FLVector'
object[pSet = 1:length(object)]
```

Arguments

`object`	is a FLVector object
`pSet`	is a vector representing the indices of elements to extract

Value

`[]` returns FLVector object after extraction which replicates the equivalent R extraction.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
WideTable <- FLTable(connection, "FL_DEMO", "tblAbaloneWide","ObsID")
flvector <- FLVector[,"Diameter"]
resultFLVector <- flvector[10:1]
```

`%%` *remainder of division on in-database objects.*

Description

`%%` calculates the remainder of in-database object division.

Usage

```
pObj1 %% pObj2
```

Arguments

`x`	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
`y`	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The remainder of in-database objects mimics the normal remainder of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

`%%` returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Constraints

division by 0 gives inf in R, but is not supported for in-database objects

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 1,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
ResultFLmatrix <- flmatrix %% Rvector
```

`%/%` *Integer Division of in-database objects.*

Description

`%/%` does the Element-wise Integer Division of in-database objects.

Usage

```
pObj1 %/% pObj2
```

Arguments

`x`	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
`y`	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Element-wise Integer Division of in-database objects mimics the `%/%` of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

`%/%` returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Constraints

division by 0 gives inf in R, but is not supported for in-database objects

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 1,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
ResultFLmatrix <- flmatrix %/% Rvector
```

%*% *Cross-Product of in-database objects.*

Description

%*% does the Cross-Product of in-database objects.

Usage

```
pObj1 %*% pObj2
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Cross-Product of in-database objects mimics the %*% of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

%*% returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_DEMO", "tblMatrixMulti", 5,"MATRIX_ID","ROW_ID","COL_ID","CELL_VAL")
Rvector <- 1:5
ResultFLmatrix <- flmatrix %*% Rvector
```

Index