Package 'dbMatrix'

May 19, 2026

Title Database-Backed Matrix Classes and Operations

Version 0.1.0

Description Provides S4 classes and methods for storing dense and sparse matrices in 'DuckDB' databases. The package supports constructing database-backed matrices from base R and 'Matrix' objects, extracting slices and summaries, performing arithmetic and selected linear algebra operations, and materializing results for larger-than-memory workflows. It integrates with 'dbProject' to keep database paths, live connections, and lazy matrix tables synchronized across interactive analyses.

License GPL-3

Encoding UTF-8

URL https://github.com/dbverse-org/dbmatrix-r, https://dbverse-org.github.io/dbmatrix-r/

BugReports https://github.com/dbverse-org/dbmatrix-r/issues

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports Matrix (>= 1.6-5), MatrixGenerics (>= 1.12.3), methods, DBI, dplyr, dbplyr, duckdb (>= 1.4.0), data.table (>= 1.12.2), glue, bit64, cli, Rcpp, arrow, nanoarrow, dbProject, rlang

LinkingTo Rcpp, RcppEigen, RSpectra, nanoarrow

Suggests knitr, rmarkdown, testthat (>= 3.0.0), irlba, crayon, R.utils, checkmate, reticulate, sparseMatrixStats, RSpectra

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Edward C. Ruiz [aut, cre] (ORCID: <https://orcid.org/0000-0002-9174-5387>), Jiaji George Chen [aut], Ruben Dries [aut]

Maintainer Edward C. Ruiz <ecr7407@gmail.com>

Repository CRAN

Date/Publication 2026-05-19 08:30:02 UTC

Contents

`Arith,dbMatrix,ANY-method`

Arith dbMatrix, e2

Description

See `methods::Arith` for more details.

See `methods::Arith` for more details.

See `methods::Arith` for more details.

See `methods::Ops` for more details.

See `methods::Ops` for more details.

See `methods::Ops` for more details.

Usage

```
## S4 method for signature 'dbMatrix,ANY'
Arith(e1, e2)

## S4 method for signature 'ANY,dbMatrix'
Arith(e1, e2)

## S4 method for signature 'dbMatrix,dbMatrix'
Arith(e1, e2)

## S4 method for signature 'dbMatrix,ANY'
Ops(e1, e2)

## S4 method for signature 'ANY,dbMatrix'
Ops(e1, e2)

## S4 method for signature 'dbMatrix,dbMatrix'
Ops(e1, e2)

## S4 method for signature 'DBIConnection'
dbLoad(conn, name, class)

## S4 method for signature 'dbMatrix'
writeMM(obj, file, ...)
```

Arguments

`e1`	First operand.
`e2`	Second operand.
`conn`	DBIConnection object
`name`	valid name value (character)
`class`	character, class of the dbMatrix object (e.g. "dbDenseMatrix" or "dbSparseMatrix")
`obj`	dbMatrix object
`file`	path to file
`...`	additional arguments

Value

- Arithmetic and logical group methods return a dbMatrix object of the appropriate dense or sparse subclass, with the same dimensions as the input and transformed values stored in DuckDB.
- dbLoad() returns a dbDenseMatrix or dbSparseMatrix pointing to an existing DuckDB table.
- writeMM() writes a Matrix Market file to file and returns invisible(TRUE) on success.

---

as.dbMatrix *Convert* Matrix::Matrix *to* dbMatrix

Description

Converts in-memory matrix, Matrix::dgeMatrix, or Matrix::dgCMatrix into a dbMatrix object.

Generic function to convert in-memory objects to dbMatrix objects.

Usage

```
as.dbMatrix(x, con = NULL, name = "dbMatrix", overwrite = FALSE, ...)

as.dbMatrix(x, con = NULL, name = "dbMatrix", overwrite = FALSE, ...)
```

Arguments

x	Object to convert (e.g., matrix, dgCMatrix)
con	DBI or duckdb connection object
name	Table name to assign within database
overwrite	Whether to overwrite if table already exists
...	Additional arguments passed to methods

Details

If no con is provided, a temporary in-memory database connection is created. If no name is provided, a unique table name is generated.

Value

A dbDenseMatrix for dense matrix inputs or a dbSparseMatrix for sparse matrix inputs. The returned object keeps the input dimensions and dimnames while storing matrix values in DuckDB.

as.matrix.dbMatrix *Convert dbMatrix to in-memory matrix*

Description

Converts a dbMatrix object into an in-memory matrix or sparse matrix.

Usage

```
## S3 method for class 'dbMatrix'
as.matrix(x, ..., sparse = FALSE, names = TRUE)
```

Arguments

x	A dbMatrix object (dbSparseMatrix or dbDenseMatrix)
...	Additional arguments (not used)
sparse	Logical indicating if the output should be a sparse matrix default:FALSE
names	Logical indicating if the output should have dimnames. default:FALSE

Details

This method converts a dbMatrix object into an in-memory Matrix::dgCMatrix (sparse = TRUE) or matrix() (default, sparse = FALSE).

Warning: This function can cause memory issues for large dbMatrix objects.

Set sparse = TRUE to convert to a sparse matrix. Set names = TRUE to keep dimnames.

Value

A Matrix::dgCMatrix or matrix

coerce-dbMatrix-dgCMatrix *Coerce dbMatrix to dgCMatrix*

Description

Coercion methods to convert dbMatrix objects to in-memory dgCMatrix objects. Respects dbMatrix.max_mem_convert option to prevent OOM errors.

Value

A Matrix::dgCMatrix object containing the collected matrix values. Dense inputs are converted to sparse Matrix format after collection.

`coerce-dbMatrix-matrix`

Coerce dbMatrix to matrix

Description

Coercion methods to convert `dbMatrix` objects to in-memory `matrix` objects. Respects `dbMatrix.max_mem_convert` option to prevent OOM errors.

Value

A base R `matrix` containing the collected matrix values with the same dimensions and dimnames as the source object.

`coerce-matrix-dbMatrix`

Coerce matrix to dbMatrix

Description

Coercion methods to convert in-memory `matrix` objects to `dbMatrix` objects. Creates a new in-memory DuckDB connection.

Value

A database-backed matrix object. Dense inputs return a `dbDenseMatrix`, while sparse `Matrix::dgCMatrix` inputs return a `dbSparseMatrix`.

`colSds,dbDenseMatrix-method`

Row (column) standard deviations for `dbMatrix` objects

Description

Calculates the standard deviation for each row (column) of a matrix-like object.

Usage

```
## S4 method for signature 'dbDenseMatrix'
colSds(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  center = NULL,
  ...,
  useNames = TRUE
)

## S4 method for signature 'dbSparseMatrix'
colSds(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  center = NULL,
  ...,
  useNames = TRUE
)

## S4 method for signature 'dbDenseMatrix'
rowSds(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = TRUE,
  center = NULL,
  ...,
  useNames = TRUE
)

## S4 method for signature 'dbSparseMatrix'
rowSds(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = TRUE,
  center = NULL,
  ...,
  useNames = TRUE
)
```

Arguments

x	A dbMatrix object.

rows	Always NULL for dbMatrix queries. TODO
cols	Always NULL for dbMatrix queries. TODO
na.rm	Always TRUE for dbMatrix queries. Included for compatibility with the generic.
center	Always NULL for dbMatrix queries. Included for compatibility with the generic.
...	Additional arguments (not used, but included for compatibility with the generic).
useNames	Always TRUE for dbMatrix queries. Included for compatibility with the generic.

Value

A named numeric vector containing one sample standard deviation per row or column of x.

---

compute.dbMatrix *Force computation of a dbMatrix*

Description

Explicitly compute a dbMatrix and save it to a table in the database. This overrides the default dplyr::compute to use a direct CREATE TABLE AS statement, which is more robust for large tables in DuckDB.

Usage

```
## S3 method for class 'dbMatrix'
compute(
  x,
  name = NULL,
  temporary = TRUE,
  dimnames = TRUE,
  overwrite = FALSE,
  ...
)
```

Arguments

x	A dbMatrix object
name	Name of the table to create. If NULL, a random name is generated.
temporary	Logical. If TRUE (default), create a temporary table.
dimnames	default = TRUE. If TRUE, the rownames and colnames will be saved in the database. This allows full reconstruction of the dbMatrix object using dbProject::dbLoad().
overwrite	Logical. If TRUE, overwrite the table if it already exists. Default is FALSE.
...	Additional arguments passed to methods (ignored).

Value

A dbMatrix object pointing to the new table.

dbDenseMatrix-class *S4 Class for* dbDenseMatrix

Description

Representation of dense matrices using an on-disk database. Inherits from dbMatrix.

Value

Objects of class dbDenseMatrix store all matrix entries explicitly in DuckDB. They are typically returned by dbMatrix() or as.dbMatrix() for dense inputs.

dbMatrix_from_tbl *dbMatrix_from_tbl*

Description

Constructs a dbSparseMatrix object from a tbl_duckdb_connection object.

Usage

```
dbMatrix_from_tbl(
  tbl,
  rownames_colName,
  colnames_colName,
  value_colName = NULL,
  name = "dbMatrix",
  overwrite = FALSE,
  row_names = NULL,
  col_names = NULL,
  i_col = NULL,
  j_col = NULL
)
```

Arguments

tbl	tbl_duckdb_connection table in DuckDB database in long format
rownames_colName	character column name of rownames in tbl (required)
colnames_colName	character column name of colnames in tbl (required)
value_colName	character column name containing pre-aggregated integer counts. If NULL (default), counts occurrences of each row-column pair. (optional)
name	table name to assign within database (required, default: "dbMatrix")

overwrite	whether to overwrite if table already exists in database (required)
row_names	character vector of pre-computed row names (sorted). If NULL (default), row names are extracted from the table. (optional)
col_names	character vector of pre-computed column names (sorted). If NULL (default), column names are extracted from the table. (optional)
i_col	character column name containing pre-computed row indices (1-based integers). If provided with j_col, skips index encoding for optimal performance. (optional)
j_col	character column name containing pre-computed column indices (1-based integers). If provided with i_col, skips index encoding for optimal performance. (optional)

Details

The tbl_duckdb_connection object must contain dimension names as columns in long format.

If value_colName is provided, the function uses pre-aggregated counts from that column. This is useful when the input table already contains aggregated counts (e.g., from a GROUP BY + SUM operation). If value_colName is NULL (default), the function counts occurrences of each row-column pair.

When row_names and/or col_names are provided, the function uses these directly instead of querying distinct values from the table. This can significantly improve performance when the input table is a complex lazy query (e.g., result of spatial joins).

When i_col and j_col are provided, the function uses these pre-computed integer indices directly, skipping expensive string-to-index encoding. This is the fastest path.

Value

dbMatrix object

---

dbMatrix_options *dbMatrix Package Global Options*

Description

The following global options can be modified to control the behavior of the dbMatrix package.

Details

Use options() to set the below options.

Value

No return value. This documentation page describes package options.

Options

- `dbMatrix.digits`: integer. Number of digits to round to in the show function of dbMatrix objects. Default is 7.
- `dbMatrix.max_mem_convert`: numeric. Maximum size (in bytes) allowed for implicit conversion of `dbMatrix` to in-memory matrix. Default is 8 * 1024^3 (8GB).
- `dbMatrix.chunk_size`: integer. Number of columns to process per chunk during densification (`.to_db_dense`). Smaller chunks reduce memory usage but may increase execution time. Default is `NULL` (automatically calculated based on `dbMatrix.max_mem_convert`).
- `dbMatrix.max_chunks`: integer. Maximum number of chunks allowed during densification. This prevents query parser errors caused by excessive `UNION ALL` branches. Default is 10000.
- `dbMatrix.verbose`: logical. If `TRUE` (default), prints informative messages during implicit coercion.
- `dbMatrix.precomp_db`: character. Path to an external DuckDB file containing precomputed table(s). If set, `dbMatrix` will automatically attach this database (read-only) and look for a suitable precomputed table to speed up densification.
- `dbMatrix.allow_densify`: logical. If `FALSE` (default), automatic sparse-to-dense conversion is disabled. This prevents unexpected disk spilling and memory issues when operations would require densification (e.g., division by zero, scalar addition to sparse matrix). Set to `TRUE` to enable on-disk dense conversion. **Warning**: Dense conversion can cause massive disk usage for large matrices.

`dbSparseMatrix-class` *S4 Class for dbSparseMatrix*

Description

Representation of sparse matrices using an on-disk database. Inherits from dbMatrix.

Value

Objects of class `dbSparseMatrix` store only non-zero matrix entries in DuckDB. They are typically returned by `dbMatrix()` or `as.dbMatrix()` for sparse inputs.

`db_svd` *Perform Streaming SVD on a dbMatrix*

Description

Perform Streaming SVD on a dbMatrix

Usage

```
db_svd(
  dbm,
  k = 10,
  center = TRUE,
  scale = FALSE,
  center_rows = NULL,
  memory_limit = getOption("dbMatrix.svd_memory", 8 * 1024^3),
  return_format = c("svd", "pca")
)
```

Arguments

dbm	A dbSparseMatrix object
k	Number of singular values to compute
center	Logical, center rows (default TRUE)
scale	Logical, scale rows (default FALSE)
center_rows	Logical, center rows vs columns (default TRUE for standard PCA)
memory_limit	Bytes for Fast Path. Default 8 GB.
return_format	"svd" (d, u, v) or "pca" (eigenvalues, loadings, coords)

Value

List with SVD or PCA components

---

dim,dbMatrix-method *Dimensions of an Object*

Description

Retrieve the dimension of an object.

Usage

```
## S4 method for signature 'dbMatrix'
dim(x)
```

Arguments

x	dbMatrix object

Value

An integer vector of length 2 giving the number of rows and columns in x.

head,dbMatrix-method *Return the First or Last Parts of an Object*

Description

Returns the first or last parts of a vector, matrix, array, table, data frame or function. Since `head()` and `tail()` are generic functions, they have been extended to other classes, including "`ts`" from **stats**.

Usage

```
## S4 method for signature 'dbMatrix'
head(x, n = 6L, ...)

## S4 method for signature 'dbMatrix'
tail(x, n = 6L, ...)
```

Arguments

`x`	an object
`n`	an integer vector of length up to `dim(x)` (or 1, for non-dimensioned objects). A `logical` is silently coerced to integer. Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of `n[i]` includes the first/last `n[i]` indices in that dimension, while a negative value excludes the last/first `abs(n[i])`, including all remaining indices. `NA` or non-specified values (when `length(n) < length(dim(x))`) select all indices in that dimension. Must contain at least one non-missing value.
`...`	arguments to be passed to or from other methods.

Value

A `dbMatrix` object containing the first or last `n` rows of `x`, with updated dimensions and row names.

is.na,dbMatrix-method *Element-wise is.na for dbMatrix*

Description

Returns a dbMatrix with numeric values indicating NA positions (1 = NA, 0 = not NA).

Usage

```
## S4 method for signature 'dbMatrix'
is.na(x)
```

Arguments

x	A dbMatrix object.

Value

A dbMatrix with same dimensions, containing 1 where the original value was NA and 0 otherwise.

Examples

```
mat <- matrix(c(1, NA, 3, NA), nrow = 2)
dbmat <- as.dbMatrix(mat)
is.na(dbmat)
```

---

length,dbMatrix-method

Length of a dbMatrix *Object*

Description

Get or set the length of vectors (including lists) and factors, and of any other R object for which a method has been defined.

Usage

```
## S4 method for signature 'dbMatrix'
length(x)
```

Arguments

x	dbMatrix object

Value

A length-one integer giving the number of stored elements in x.

`Math,dbMatrix-method` *Math Operations for* `dbMatrix` *Objects*

Description

Implements the `Math` `S4groupGeneric` functions for `dbMatrix` objects. This includes various mathematical operations such as logarithms, exponentials, trigonometric functions, and other transformations.

Usage

```
## S4 method for signature 'dbMatrix'
Math(x)
```

Arguments

x	A `dbMatrix` object.

Details

This method provides implementations for the following Math functions:

Arithmetic and rounding:

- `abs()`, `sign()`, `sqrt()`, `ceiling()`, `floor()`, `trunc()`

Cumulative operations:

- `cummax()`, `cummin()`, `cumprod()`, `cumsum()`
- **Note:** `cumprod()` **is not supported**

Logarithmic:

- `log()`, `log10()`, `log2()`, `log1p()`

DuckDB Log Function Mappings:

R Function	DuckDB Function	Notes
`log(x)`	`LN(x)`	Natural logarithm
`log10(x)`	`LOG10(x)`	Base-10 logarithm
`log2(x)`	`LOG2(x)`	Base-2 logarithm
`log1p(x)`	`LN(x + 1)`	log(1+x), computed as LN

Sparsity-Preserving Log: For `dbSparseMatrix` with pending operations, `log(x + 1)` operations preserve sparsity since `log(0 + 1) = 0`. The multiplicative component is applied first, then the log transformation is applied to sparse values only.

Trigonometric:

- `cos()`, `sin()`, `tan()`, `acos()`, `asin()`, `atan()`
- `cosh()`, `sinh()`, `tanh()`, `acosh()`, `asinh()`, `atanh()`
- `cospi()`, `sinpi()`, `tanpi()`
- **Note:** `acosh()` `asinh()` `atanh()` **are not supported**

Exponential:

- `exp()`, `expm1()`
- **Note:** `expm1()` **is not supported**

Special functions:

- `gamma()`, `lgamma()`, `digamma()`, `trigamma()`
- **Note:** `digamma()` `trigamma()` **are not supported**

The function applies the specified mathematical operation to each element of the `dbMatrix` object.

Value

A `dbMatrix` object with the mathematical operation applied to each element.

Examples

```
mat <- matrix(1, nrow = 3, ncol = 3)
dbmat <- as.dbMatrix(mat)
log(dbmat)
sqrt(dbmat)
sin(dbmat)
```

---

mean,dbDenseMatrix-method

Arithmetic Mean for `dbMatrix` *objects*

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
## S4 method for signature 'dbDenseMatrix'
mean(x, ...)

## S4 method for signature 'dbSparseMatrix'
mean(x, ...)
```

Arguments

x	dbMatrix object
...	further arguments passed to or from other methods.

Value

A length-one numeric vector giving the arithmetic mean of all entries in x.

---

names,dbDenseMatrix-method *The names of a dbMatrix Object*

Description

The names of a dbMatrix Object

Usage

```
## S4 method for signature 'dbDenseMatrix'
names(x)
```

Arguments

x	A dbMatrix object

Value

A character vector of the names of the 1D dbMatrix object (1D matrices only)

---

nrow.dbMatrix *The Number of Rows/Columns of a dbMatrix Object*

Description

nrow and ncol return the number of rows or columns present in x.

Usage

```
nrow.dbMatrix(x)

ncol.dbMatrix(x)
```

Arguments

x	dbMatrix object

Value

A length-one integer giving the number of rows or columns in x.

`rowMeans,dbMatrix-method` *Row (column) means for* `dbMatrix` *objects*

Description

Calculates the mean for each row (column) of a matrix-like object.

Usage

```
## S4 method for signature 'dbMatrix'
rowMeans(x, na.rm = FALSE, dims = 1, ...)

## S4 method for signature 'dbMatrix'
colMeans(x, na.rm = FALSE, dims = 1, ...)
```

Arguments

x	An NxK matrix-like object, a numeric data frame, or an array-like object of two or more dimensions.
na.rm	Always TRUE for `dbMatrix` queries. Included for compatibility with the generic.
dims	Always 1 for `dbMatrix` queries. Included for compatibility with the generic.
...	Additional arguments passed to specific methods.

Value

A named numeric vector containing one mean per row or column of x.

`rownames.dbMatrix` *Retrieve and Set Row (Column) Dimension Names of dbMatrix Objects*

Description

Retrieve and Set Row (Column) Dimension Names of dbMatrix Objects

Usage

```
rownames.dbMatrix(x, do.NULL = TRUE, prefix = "row")

## S3 replacement method for class 'dbMatrix'
rownames(x) <- value

colnames.dbMatrix(x, do.NULL = TRUE, prefix = "col")

## S3 replacement method for class 'dbMatrix'
colnames(x) <- value

## S4 method for signature 'dbMatrix'
dimnames(x)

## S4 replacement method for signature 'dbMatrix,list'
dimnames(x) <- value
```

Arguments

x	a matrix-like R object, with at least two dimensions for colnames.
do.NULL	Not used for this method. Included for compatibility with the generic.
prefix	Not used for this method. Included for compatibility with the generic.
value	a valid value for that component of dimnames(x). For a matrix or array this is either NULL or a character vector of non-zero length equal to the appropriate dimension.

Value

rownames() and colnames() return character vectors of dimension names. dimnames() returns a length-2 list containing row and column name vectors. The replacement forms return the modified dbMatrix object.

---

rowSums,dbDenseMatrix-method

Row (column) sums for dbMatrix objects

Description

Calculates the sum for each row (column) of a matrix-like object.

Usage

```
## S4 method for signature 'dbDenseMatrix'
rowSums(x, na.rm = FALSE, dims = 1, ..., memory = FALSE)

## S4 method for signature 'dbSparseMatrix'
```

```
rowSums(x, na.rm = FALSE, dims = 1, ...)

## S4 method for signature 'dbDenseMatrix'
colSums(x, na.rm = FALSE, dims = 1, ...)

## S4 method for signature 'dbSparseMatrix'
colSums(x, na.rm = FALSE, dims = 1, ...)
```

Arguments

x	An NxK matrix-like object, a numeric data frame, or an array-like object of two or more dimensions.
na.rm	Always TRUE for dbMatrix. Included for compatibility with the generic.
dims	Always 1 for dbMatrix queries. Included for compatibility with the generic.
...	Additional arguments passed to specific methods.
memory	logical. If FALSE (default), results returned as dbDenseMatrix. This is recommended for large computations. Set to TRUE to return the results as a vector.

Value

A named numeric vector containing one sum per row or column of x.

---

rowVars,dbDenseMatrix-method

Row (column) variances for dbMatrix objects

Description

Calculates the variance for each row (column) of a matrix-like object.

Usage

```
## S4 method for signature 'dbDenseMatrix'
rowVars(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = TRUE,
  center = NULL,
  ...,
  useNames = TRUE
)

## S4 method for signature 'dbSparseMatrix'
rowVars(
  x,
```

```
  rows = NULL,
  cols = NULL,
  na.rm = TRUE,
  center = NULL,
  ...,
  useNames = TRUE
)

## S4 method for signature 'dbDenseMatrix'
colVars(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = TRUE,
  center = NULL,
  ...,
  useNames = TRUE
)

## S4 method for signature 'dbSparseMatrix'
colVars(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = TRUE,
  center = NULL,
  ...,
  useNames = TRUE
)
```

Arguments

x	A dbMatrix object.
rows	Always NULL for dbMatrix queries. Included for compatibility with the generic.
cols	Always NULL for dbMatrix queries. Included for compatibility with the generic.
na.rm	Always TRUE for dbMatrix queries. Included for compatibility with the generic.
center	Always NULL for dbMatrix queries. Included for compatibility with the generic.
...	Additional arguments (not used, but included for compatibility with the generic).
useNames	Always TRUE for dbMatrix queries. Included for compatibility with the generic.

Value

A named numeric vector containing one sample variance per row or column of x.

Summary,dbMatrix-method

Summary Methods for dbMatrix *Objects*

Description

Implements the S4groupGeneric group generic functions for dbMatrix objects.

Usage

```
## S4 method for signature 'dbMatrix'
Summary(x, ..., na.rm = TRUE)
```

Arguments

x	A dbMatrix object.
...	Additional arguments (not used, but included for compatibility with the generic).
na.rm	Logical. If TRUE, remove NA values before computation. Always set to TRUE for this implementation.

Details

This method provides implementations for the following S4groupGeneric functions:

- max(): Maximum value
- min(): Minimum value
- range(): *Not supported*
- prod(): Product of all values
- sum(): Sum of all values
- any(): Returns TRUE if any value is TRUE
- all(): Returns TRUE if all values are TRUE

Value

The result of applying the respective summary function to the dbMatrix object. The type of the return value depends on the specific function called.

Examples

```
mat <- matrix(1, nrow = 3, ncol = 3)
dbmat <- as.dbMatrix(mat)
max(dbmat)
min(dbmat)
prod(dbmat)
sum(dbmat)
any(dbmat > 0)
all(dbmat > 0)
```

t,dbMatrix-method *Matrix Transpose*

Description

Given a dbMatrix x, t returns the transpose of x.

Usage

```
## S4 method for signature 'dbMatrix'
t(x)
```

Arguments

x	dbMatrix object

Value

dbMatrix object

to_named_ijx_tbl *Convert dbMatrix to named ijx table*

Description

Converts a dbMatrix to a lazy long table where row and column indices are replaced by dimension names.

Usage

```
to_named_ijx_tbl(
  x,
  row_col = "row_name",
  col_col = "col_name",
  compute = FALSE
)
```

Arguments

x	A dbMatrix object (dbSparseMatrix or dbDenseMatrix)
row_col	Name for the row-name column (default: "row_name")
col_col	Name for the column-name column (default: "col_name")
compute	Whether to materialize as temp table (default: FALSE)

Value

A lazy tbl with columns: row_col, col_col, x

`[,dbMatrix,dbIndex,missing,ANY-method`
Extract or replace values in database-backed matrices

Description

Methods for subsetting and replacing values in `dbMatrix` objects.

Usage

```
## S4 method for signature 'dbMatrix,dbIndex,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'dbMatrix,missing,dbIndex,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'dbMatrix,dbIndex,dbIndex,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'dbMatrix,dbMatrix,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'dbMatrix,dbMatrix,missing,ANY'
x[i, j] <- value

## S4 method for signature 'dbMatrix,dbDenseMatrix,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'dbMatrix,missing,dbDenseMatrix,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'dbMatrix,dbDenseMatrix,dbDenseMatrix,ANY'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	A `dbMatrix` object.
`i`	Row, logical matrix, or matrix-style index.
`j`	Column index.
`...`	Additional arguments.
`drop`	Ignored; included for matrix API compatibility.
`value`	Replacement value.

Value

A subsetted or modified `dbMatrix`, or an extracted vector for matrix-style indexing.

`%in%,dbDenseMatrix,ANY-method` *Value Matching*

Description

Implements the `%in%` operator for dbMatrix objects. This operator checks if elements from the left operand are contained in the right operand, returning a logical vector.

Usage

```
## S4 method for signature 'dbDenseMatrix,ANY'
x %in% table

## S4 method for signature 'ANY,dbDenseMatrix'
x %in% table

## S4 method for signature 'dbSparseMatrix,ANY'
x %in% table
```

Arguments

`x`	A dbMatrix object or any other object
`table`	Any object or a dbMatrix object

Details

This is a method for the standard `%in%` operator for dbMatrix objects. It follows R's standard behavior for the `%in%` operator:

- When `x` is a dbDenseMatrix, it returns a logical vector with the same length as the total number of elements in the matrix.
- When `table` is a dbDenseMatrix, it allows checking if elements in `x` are in the matrix.
- For dbSparseMatrix objects, it throws an error to match the behavior of dgCMatrix.

Value

A logical vector of the same length as `x`, indicating which elements of `x` are in `table`.

Examples

```
con <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
mat <- matrix(1:9, nrow = 3, ncol = 3)
dbmat <- dbMatrix(
  value = mat,
  con = con,
  name = "example_matrix",
```

```
  class = "dbDenseMatrix",
  overwrite = TRUE
)

dbmat %in% c(1, 3, 5, 7, 9)

c(1, 3, 5, 7, 9) %in% dbmat
DBI::dbDisconnect(con, shutdown = TRUE)
```

Index