Package 'dbSpatial'

October 23, 2026

Title Spatial Data Operations for Database-Backed Geometries

Version 0.1.4

Description Provides database-backed spatial geometry classes and methods for working with vector spatial data in 'DuckDB'. The package supports loading, converting, querying, joining, and measuring spatial geometries through familiar 'sf'-style interfaces while keeping geometry columns lazy inside the database. It integrates with 'dbProject' to preserve database paths, live connections, and spatial table metadata across interactive sessions. The package follows the Simple Features framework described by Pebesma (2018) <doi:10.32614/RJ-2018-009> and uses DuckDB's spatial extension <https://duckdb.org/docs/stable/core_extensions/spatial/overview.html>.

License GPL-3 | MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports methods, tools, DBI, duckdb (>= 1.4.0), dplyr, dbplyr, terra, sf, e1071, lifecycle, crayon, checkmate, cli, rlang, glue, tidyselect, dbProject

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

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

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

Config/testthat/edition 3

Collate 'utils.R' 'generics.R' 'classes.R' 'accessors.R' 'as_dbSpatial.R' 'coerce.R' 'constructors.R' 'dbSpatial.R' 'extract.R' 'geometry_ops.R' 'global.R' 'imports.R' 'input_validation.R' 'loadSpatial.R' 'measurements.R' 'show.R' 'sim_dbSpatial.R' 'sql_gen.R' 'st_as_sf.R' 'st_bbox.R' 'st_geometry_type.R' 'st_geometrytype.R' 'st_is_valid.R' 'st_spatial_join.R' 'st_join.R' 'st_read.R' 'st_translate.R' 'st_xmax.R' 'st_ymax.R' 'vect.R'

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-10-23 18:00:02 UTC

Contents

as_dbSpatial . . . 3
dbSpatial . . . 4
dbSpatial_options . . . 5
head,dbSpatial-method . . . 6
loadSpatial . . . 7
show,dbSpatial-method . . . 7
st_area . . . 8
st_as_geojson . . . 9
st_as_sf.dbSpatial . . . 10
st_as_text . . . 11
st_bbox . . . 12
st_buffer . . . 12
st_centroid . . . 13
st_geometrytype . . . 14
st_join.dbSpatial . . . 15
st_length . . . 16
st_npoints . . . 17
st_perimeter . . . 18
st_simplify . . . 19
st_translate . . . 20
st_x . . . 21
st_xmax . . . 22
st_y . . . 23
st_ymax . . . 24
tail,dbSpatial-method . . . 25
vect,dbSpatial-method . . . 26
$,dbSpatial-method . . . 27

Index **28**

as_dbSpatial *Convert an sf or terra object to a* dbSpatial *object*

Description

Create a dbSpatial object from an sf or terra object.

Usage

```
as_dbSpatial(rSpatial, conn, name, overwrite = FALSE, temporary = TRUE, ...)
```

Arguments

rSpatial	sf or terra object.
conn	A DBI connection object, as returned by DBI::dbConnect().
name	a character string with the unquoted DBMS table name, e.g. "table_name"
overwrite	logical. Overwrite existing table. default = FALSE.
temporary	logical. Create a temporary table. default = TRUE.
...	Additional arguments to be passed

Details

Writes the rSpatial object to a temporary DuckDB table and computes the table in the database with the specified name and the geometry column as geom.

Value

A dbSpatial object backed by table name in conn, with the geometry column stored as geom.

See Also

Other dbSpatial: dbSpatial, show,dbSpatial-method, st_as_sf.dbSpatial(), vect,dbSpatial-method

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  coordinates <- data.frame(x = c(100, 200, 300), y = c(500, 600, 700))
  attributes <- data.frame(id = 1:3, name = c("A", "B", "C"))

  # Combine the coordinates and attributes
  dummy_data <- cbind(coordinates, attributes)

  # Create a SpatVector from the data.frame
  dummy_spatvector <- terra::vect(dummy_data, geom = c("x", "y"))

  # Set db connection
  duckdb_conn = DBI::dbConnect(duckdb::duckdb(), ":memory:")
```

```
  DBI::dbExecute(duckdb_conn, "SET threads = 1")

  dbSpatial <- as_dbSpatial(rSpatial = dummy_spatvector,
                            conn = duckdb_conn,
                            name = "dummy_spatvector",
                            overwrite = TRUE)
  dbSpatial
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

dbSpatial *Create a* dbSpatial *object with geometry data type*

Description

Constructor function to ingest diverse spatial data sources and create a dbSpatial object containing a geometry data type based on the Simple Features standard.

If x_colName and y_colName are both provided, a POINT geometry will be constructed based on these columns.

Usage

```
dbSpatial(
  value,
  name,
  conn,
  x_colName = NULL,
  y_colName = NULL,
  geomName = "geom",
  overwrite = FALSE,
  ...
)
```

Arguments

value	data.frame, tbl_duckdb_connection, character (valid file path), sf object, or terra object. Data to construct dbSpatial object with geometry data type. See details for more information.
name	Table name.
conn	A DBI connection object.
x_colName	character. Name of column containing numerical X coordinates. default = NULL.
y_colName	character. Name of column containing numerical Y coordinates. default = NULL.
geomName	character string. The geometry column name in the dbSpatial object. Default: "geom".
overwrite	logical. Overwrite existing table. default = FALSE.
...	Additional arguments to be passed

Details

For list of files supported see link below. https://DuckDB.org/docs/extensions/spatial.html#st_read---read-spatial-value-from-files

Value

dbSpatial object.

See Also

Other dbSpatial: as_dbSpatial(), show,dbSpatial-method, st_as_sf.dbSpatial(), vect,dbSpatial-method

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  # create in-memory DuckDB db
  duckdb_conn = DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")

  # test value
  test_data = data.frame(x = 1:10, y = 1:10, id = 1:10)

  test_file <- tempfile(fileext = ".csv")
  write.csv(test_data, test_file, row.names = FALSE)

  # read data.frame and create point geometry
  dbSpatial(conn = duckdb_conn,
            name = "test_points",
            value = test_data,
            x_colName = "x",
            y_colName = "y",
            overwrite = TRUE)

  # read csv
  dbSpatial(conn = duckdb_conn,
            name = "test_points",
            value = test_file,
            x_colName = "x",
            y_colName = "y",
            overwrite = TRUE)
  unlink(test_file)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

dbSpatial_options *dbSpatial Package Global Options*

Description

Global options to control package behavior.

Details

Use `options()` to set the below options.

Options

- `dbSpatial.max_print`: integer. Max characters for WKT in show method (default 30).
- `dbSpatial.max_mem_convert`: numeric. Max bytes for implicit coercion (default 8GB).
- `dbSpatial.verbose`: logical. Print info messages during coercion (default TRUE).
- `dbSpatial.extension_directory`: character. Directory used by DuckDB to install extensions in `loadSpatial()` (default: a directory under `tempdir()`).

Examples

```
options(dbSpatial.max_print = 50)
options(dbSpatial.max_mem_convert = 16 * 1024^3)
options(dbSpatial.verbose = FALSE)
options(dbSpatial.extension_directory = file.path(tempdir(), "duckdb-extensions"))
```

`head,dbSpatial-method` *head method for dbSpatial*

Description

head method for dbSpatial

Usage

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

Arguments

`x`	A `dbSpatial` object.
`n`	Number of rows to return.
`...`	Additional arguments.

Value

A `dbSpatial` object containing the first `n` rows of `x`.

See Also

Other dbData: `tail,dbSpatial-method`

loadSpatial *Install and/or load DuckDB spatial extension*

Description

Install and/or load DuckDB spatial extension

Usage

```
loadSpatial(
  conn,
 extension_directory = getOption("dbSpatial.extension_directory", file.path(tempdir(),
    "dbSpatial-duckdb-extensions"))
)
```

Arguments

conn	duckdb connection
extension_directory	Directory where DuckDB should install extensions. Defaults to a directory in the R session temporary directory. Set the `dbSpatial.extension_directory` option to use a different location.

Value

No return value, called for side effects. Installs and loads the DuckDB spatial extension for `conn`.

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  duckdb_conn = DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  loadSpatial(conn = duckdb_conn)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

show,dbSpatial-method *show method for dbSpatial*

Description

Show method for dbSpatial

Usage

```
## S4 method for signature 'dbSpatial'
show(object)
```

Arguments

object	A dbSpatial object.

See Also

Other dbSpatial: as_dbSpatial(), dbSpatial, st_as_sf.dbSpatial(), vect,dbSpatial-method

---

st_area *Get area of geometries*

Description

Returns the area of the geometry column.

Usage

```
## S3 method for class 'dbSpatial'
st_area(x, ...)
```

Arguments

x	dbSpatial object
...	ignored, with a warning

Value

dbSpatial object (lazy tibble with area column)

See Also

Other measurements: st_length(), st_perimeter()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  square <- sf::st_sf(
    id = 1,
    geom = sf::st_sfc(
      sf::st_polygon(list(rbind(
        c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
      )))
    )
  )
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(square, conn = duckdb_conn, name = "square",
                    overwrite = TRUE)
  st_area(x)
```

```
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

`st_as_geojson` *Convert to GeoJSON*

Description

Returns the GeoJSON representation of the geometry.

Usage

```
st_as_geojson(x, ...)

## S3 method for class 'dbSpatial'
st_as_geojson(x, ...)
```

Arguments

`x`	`dbSpatial` object
`...`	additional arguments

Value

`dbSpatial` object with GeoJSON column

Functions

- `st_as_geojson(dbSpatial)`: Method for `dbSpatial` objects

See Also

Other constructors: `st_as_text()`

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  point <- sf::st_sf(id = 1, geom = sf::st_sfc(sf::st_point(c(1, 2))))
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(point, conn = duckdb_conn, name = "point",
                    overwrite = TRUE)
  st_as_geojson(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

st_as_sf.dbSpatial *Convert dbSpatial objects to sf objects*

Description

S3 method implementation for converting dbSpatial objects to sf objects.

Usage

```
## S3 method for class 'dbSpatial'
st_as_sf(x, geomName = "geom", select = tidyselect::everything(), ...)
```

Arguments

x	A dbSpatial object to convert
geomName	character string. The geometry column name in the dbSpatial object. Default: "geom".
select	Columns to retain in output (default: all columns)
...	Additional arguments passed to sf::st_read

Details

This method handles conversion of dbSpatial to sf::sf objects using:

- Dynamic column selection via tidyselect semantics
- Automatic geometry column preservation
- SQL-level column subsetting for efficiency

Value

An sf::sf object

See Also

Other dbSpatial: as_dbSpatial(), dbSpatial, show,dbSpatial-method, vect,dbSpatial-method

st_as_text *Convert to WKT*

Description

Returns the Well-Known Text (WKT) representation of the geometry.

Usage

```
## S3 method for class 'dbSpatial'
st_as_text(x, ...)
```

Arguments

x	dbSpatial object
...	ignored, with a warning

Value

dbSpatial object with WKT column

See Also

Other constructors: st_as_geojson()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  point <- sf::st_sf(id = 1, geom = sf::st_sfc(sf::st_point(c(1, 2))))
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(point, conn = duckdb_conn, name = "point",
                    overwrite = TRUE)
  st_as_text(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

st_bbox *Compute bounding box for dbSpatial object*

Description

Returns the bounding box of the geometry column.

Usage

```
## S3 method for class 'dbSpatial'
st_bbox(obj, ...)
```

Arguments

obj	dbSpatial object
...	additional arguments passed to methods

Value

bbox object (named numeric vector with xmin, ymin, xmax, ymax)

See Also

Other geom_summary: st_xmax(), st_ymax()

st_buffer *Compute buffer around geometry*

Description

Returns a geometry that represents all points whose distance from this Geometry is less than or equal to distance.

Usage

```
## S3 method for class 'dbSpatial'
st_buffer(
  x,
  dist,
  nQuadSegs = 30,
  endCapStyle = "ROUND",
  joinStyle = "ROUND",
  mitreLimit = 1,
  singleSide = FALSE,
  ...
)
```

Arguments

x	dbSpatial object
dist	numeric distance
nQuadSegs	integer number of segments used to approximate a quarter circle.
endCapStyle, joinStyle, mitreLimit, singleSide	accepted for compatibility with sf::st_buffer(); DuckDB ST_Buffer always uses round caps and joins, so non-default values are ignored with a warning.
...	ignored, with a warning

Value

dbSpatial object with buffered geometry

See Also

Other geometry_ops: st_centroid(), st_simplify()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  point <- sf::st_sf(id = 1, geom = sf::st_sfc(sf::st_point(c(0, 0))))
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(point, conn = duckdb_conn, name = "point",
                    overwrite = TRUE)
  st_buffer(x, dist = 1)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

st_centroid *Compute centroid of geometry*

Description

Returns the geometric center of a geometry.

Usage

```
## S3 method for class 'dbSpatial'
st_centroid(x, ..., of_largest_polygon = FALSE)
```

Arguments

x	dbSpatial object
...	ignored, with a warning
of_largest_polygon	not supported by DuckDB ST_Centroid; TRUE is ignored with a warning.

Value

dbSpatial object with centroid geometry

See Also

Other geometry_ops: st_buffer(), st_simplify()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  square <- sf::st_sf(
    id = 1,
    geom = sf::st_sfc(
      sf::st_polygon(list(rbind(
        c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
      )))
    )
  )
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(square, conn = duckdb_conn, name = "square",
                    overwrite = TRUE)
  st_centroid(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

st_geometrytype *Get geometry types (DuckDB-native)*

Description

Returns the geometry type for each row in a dbSpatial object using DuckDB's spatial function ST_GeometryType().

Usage

```
st_geometrytype(dbSpatial, geomName = "geom", collect = FALSE, n = NULL, ...)
```

Arguments

dbSpatial	A dbSpatial object.
geomName	Geometry column name. Default: "geom".
collect	Logical (default = FALSE). If TRUE, collect results into a character vector.
n	Optional integer. If provided alongside collect = TRUE, limits the query to the first n rows before collecting.
...	Additional arguments (ignored).

Value

If `collect = FALSE` (default), a lazy tibble with a single column `geom_type`. If `collect = TRUE`, a character vector of geometry type(s).

`st_join.dbSpatial` *Spatial join for dbSpatial objects*

Description

Performs a spatial join between two dbSpatial objects using a specified spatial predicate function.

Usage

```
## S3 method for class 'dbSpatial'
st_join(x, y, join = st_intersects, suffix = c(".x", ".y"), ...)
```

Arguments

`x`	A `dbSpatial` object.
`y`	A `dbSpatial` object.
`join`	Spatial predicate function to use for the join. Default is `st_intersects`. Supported predicates: `st_intersects`, `st_contains`, `st_within`, `st_covers`, `st_covered_by`, `st_crosses`, `st_disjoint`, `st_equals`, `st_touches`, `st_overlaps`.
`suffix`	Character vector of length 2. Suffixes to add to duplicate column names from x and y. Default is `c(".x", ".y")`.
`...`	Additional arguments passed to the internal join function.

Details

This function follows the `sf` pattern where spatial predicates are passed as the `join` argument.

Value

A `dbSpatial` object containing the spatial join result.

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  con <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(con, "SET threads = 1")

  df1 <- data.frame(id = 1:3, x = c(0, 10, 20), y = c(0, 10, 20))
  pts1 <- dbSpatial(conn = con, name = "pts1", value = df1,
                    x_colName = "x", y_colName = "y", overwrite = TRUE)

  df2 <- data.frame(id = 4:6, x = c(0, 15, 25), y = c(0, 15, 25))
  pts2 <- dbSpatial(conn = con, name = "pts2", value = df2,
```

```
                    x_colName = "x", y_colName = "y", overwrite = TRUE)

  # Spatial join using intersection
  result <- sf::st_join(pts1, pts2, join = sf::st_intersects)

  # Spatial join using within predicate
  result <- sf::st_join(pts1, pts2, join = sf::st_within)

  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

---

st_length *Get length of geometries*

Description

Returns the length of the geometry column.

Usage

```
## S3 method for class 'dbSpatial'
st_length(x, ...)
```

Arguments

x	dbSpatial object
...	ignored, with a warning

Value

dbSpatial object (lazy tibble with length column)

See Also

Other measurements: st_area(), st_perimeter()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  line <- sf::st_sf(
    id = 1,
    geom = sf::st_sfc(
      sf::st_linestring(rbind(c(0, 0), c(1, 1), c(2, 1)))
    )
  )
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(line, conn = duckdb_conn, name = "line",
```

```
                    overwrite = TRUE)
  st_length(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

`st_npoints` *Get number of points*

Description

Returns the number of points in a geometry.

Usage

```
st_npoints(x, ...)

## S4 method for signature 'dbSpatial'
st_npoints(x, ...)
```

Arguments

`x`	`dbSpatial` object
`...`	additional arguments

Value

`dbSpatial` object with npoints column

Methods (by class)

- `st_npoints(dbSpatial)`: Method for `dbSpatial` objects

See Also

Other accessors: `st_x()`, `st_y()`

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  line <- sf::st_sf(
    id = 1,
    geom = sf::st_sfc(sf::st_linestring(rbind(c(0, 0), c(1, 1), c(2, 1))))
  )
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(line, conn = duckdb_conn, name = "line",
                    overwrite = TRUE)
  st_npoints(x)
```

```
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

st_perimeter *Get perimeter of geometries*

Description

Returns the perimeter of the geometry column.

Usage

```
## S3 method for class 'dbSpatial'
st_perimeter(x, ...)
```

Arguments

x	dbSpatial object
...	ignored, with a warning

Value

dbSpatial object (lazy tibble with perimeter column)

See Also

Other measurements: st_area(), st_length()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  square <- sf::st_sf(
    id = 1,
    geom = sf::st_sfc(
      sf::st_polygon(list(rbind(
        c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
      )))
    )
  )
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(square, conn = duckdb_conn, name = "square",
                    overwrite = TRUE)
  st_perimeter(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

st_simplify *Simplify geometry*

Description

Returns a simplified version of the given geometry using the Douglas-Peucker algorithm.

Usage

```
## S3 method for class 'dbSpatial'
st_simplify(x, preserveTopology = FALSE, dTolerance = 0, ...)
```

Arguments

x	dbSpatial object
preserveTopology	not supported by DuckDB ST_Simplify; TRUE is ignored with a warning.
dTolerance	numeric tolerance
...	ignored, with a warning

Value

dbSpatial object with simplified geometry

See Also

Other geometry_ops: st_buffer(), st_centroid()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  line <- sf::st_sf(
    id = 1,
    geom = sf::st_sfc(
      sf::st_linestring(rbind(
        c(0, 0), c(0.5, 0.2), c(1, 0), c(1.5, 0.1), c(2, 0)
      ))
    )
  )
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(line, conn = duckdb_conn, name = "line",
                    overwrite = TRUE)
  st_simplify(x, dTolerance = 0.1)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

st_translate *Translate x, y coordinates by delta x, delta y for point geometries*

Description

This function translates point geometries by the specified delta x and delta y values.

Usage

```
st_translate(dbSpatial, geomName = "geom", dx, dy, ...)

## S4 method for signature 'dbSpatial'
st_translate(dbSpatial, geomName = "geom", dx, dy, ...)
```

Arguments

dbSpatial	dbSpatial object.
geomName	character string. The geometry column name in the dbSpatial object. Default: "geom".
dx	numeric.value to shift x coordinates by
dy	numeric. value to shift y coordinates by
...	additional arguments passed to methods

Value

dbSpatial object

Functions

- st_translate(dbSpatial): Method for dbSpatial object

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  con = DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(con, "SET threads = 1")

  coordinates <- data.frame(x = c(100, 200, 300), y = c(500, 600, 700))
  attributes <- data.frame(id = 1:3, name = c("A", "B", "C"))

  # Combine the coordinates and attributes
  dummy_data <- cbind(coordinates, attributes)

  points <- dbSpatial(conn = con,
                      name = "points",
                      value = dummy_data,
                      overwrite = TRUE,
                      x_colName = "x",
```

```
                        y_colName = "y")

  points

  points_translated <- st_translate(dbSpatial = points, dx = 100, dy = -20)

  points_translated
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

---

st_x *Get X coordinate*

Description

Returns the X coordinate of a point.

Usage

```
st_x(x, ...)

## S4 method for signature 'dbSpatial'
st_x(x, ...)
```

Arguments

x	dbSpatial object
...	additional arguments

Value

dbSpatial object with X coordinate column

Methods (by class)

- st_x(dbSpatial): Method for dbSpatial objects

See Also

Other accessors: st_npoints(), st_y()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  point <- sf::st_sf(id = 1, geom = sf::st_sfc(sf::st_point(c(1, 2))))
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(point, conn = duckdb_conn, name = "point",
                    overwrite = TRUE)
```

```
  st_x(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

st_xmax *Get maximum x coordinate*

Description

This function returns the maximum x coordinate in each geometry in the specified dbSpatial object.

Usage

```
st_xmax(dbSpatial, geomName = "geom", ...)

## S4 method for signature 'dbSpatial'
st_xmax(dbSpatial, geomName = "geom", ...)
```

Arguments

dbSpatial	dbSpatial object.
geomName	character string. The geometry column name in the dbSpatial object. Default: "geom".
...	additional arguments passed to methods

Value

numerical column vector in database

Functions

- st_xmax(dbSpatial): Method for dbSpatial object

See Also

Other geom_summary: st_bbox(), st_ymax()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  # Create a data.frame with x and y coordinates and attributes
  coordinates <- data.frame(x = c(100, 200, 300), y = c(500, 600, 700))
  attributes <- data.frame(id = 1:3, name = c("A", "B", "C"))

  # Combine the coordinates and attributes
  dummy_data <- cbind(coordinates, attributes)
```

```
  # Create a duckdb connection
  con = DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(con, "SET threads = 1")

  # Create a duckdb table with spatial points
  db_points = dbSpatial(conn = con,
                        value = dummy_data,
                        x_colName = "x",
                        y_colName = "y",
                        name = "foo",
                        overwrite = TRUE)

  st_xmax(dbSpatial = db_points)
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

`st_y` *Get Y coordinate*

Description

Returns the Y coordinate of a point.

Usage

```
st_y(x, ...)

## S4 method for signature 'dbSpatial'
st_y(x, ...)
```

Arguments

`x`	`dbSpatial` object
`...`	additional arguments

Value

`dbSpatial` object with Y coordinate column

Methods (by class)

- `st_y(dbSpatial)`: Method for `dbSpatial` objects

See Also

Other accessors: `st_npoints()`, `st_x()`

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  point <- sf::st_sf(id = 1, geom = sf::st_sfc(sf::st_point(c(1, 2))))
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  x <- as_dbSpatial(point, conn = duckdb_conn, name = "point",
                    overwrite = TRUE)
  st_y(x)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

---

st_ymax *Get maximum y coordinate*

Description

This function returns the maximum y coordinate of the geometries in the specified dbSpatial object.

Usage

```
st_ymax(dbSpatial, geomName = "geom", ...)

## S4 method for signature 'dbSpatial'
st_ymax(dbSpatial, geomName = "geom", ...)
```

Arguments

dbSpatial	dbSpatial object.
geomName	character string. The geometry column name in the dbSpatial object. Default: "geom".
...	additional arguments passed to methods

Value

numerical column vector in database

Functions

- st_ymax(dbSpatial): Method for dbSpatial object

See Also

Other geom_summary: st_bbox(), st_xmax()

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  # Create a data.frame with x and y coordinates and attributes
  coordinates <- data.frame(x = c(100, 200, 300), y = c(500, 600, 700))
  attributes <- data.frame(id = 1:3, name = c("A", "B", "C"))

  # Combine the coordinates and attributes
  dummy_data <- cbind(coordinates, attributes)

  # Create a duckdb connection
  con = DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(con, "SET threads = 1")

  # Create a duckdb table with spatial points
  db_points = dbSpatial(conn = con,
                        value = dummy_data,
                        x_colName = "x",
                        y_colName = "y",
                        name = "foo",
                        overwrite = TRUE)

  st_ymax(dbSpatial = db_points)
  DBI::dbDisconnect(con, shutdown = TRUE)
}
```

---

tail,dbSpatial-method *tail method for dbSpatial*

Description

tail method for dbSpatial

Usage

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

Arguments

x	A dbSpatial object.
n	Number of rows to return.
...	Additional arguments.

Value

A dbSpatial object containing the last n rows of x.

See Also

Other dbData: head,dbSpatial-method

`vect,dbSpatial-method` *Create SpatVector objects*

Description

Create SpatVector objects

Usage

```
## S4 method for signature 'dbSpatial'
vect(x, select = tidyselect::everything(), ...)
```

Arguments

`x`	A `dbSpatial` object to convert into a `terra::SpatVector` object
`select`	Columns to retain in output (default: all columns)
`...`	Additional arguments passed to `sf::st_as_sf`

Value

A `terra::SpatVector` containing the selected columns and geometry materialized from `x`.

See Also

Other dbSpatial: `as_dbSpatial()`, `dbSpatial`, `show,dbSpatial-method`, `st_as_sf.dbSpatial()`

Examples

```
if (interactive() && requireNamespace("duckdb", quietly = TRUE)) {
  point_data <- data.frame(x = c(100, 200), y = c(500, 600), id = 1:2)
  point_vect <- terra::vect(point_data, geom = c("x", "y"))
  duckdb_conn <- DBI::dbConnect(duckdb::duckdb(), ":memory:")
  DBI::dbExecute(duckdb_conn, "SET threads = 1")
  dbs <- as_dbSpatial(point_vect, conn = duckdb_conn, name = "point_vect",
                      overwrite = TRUE)
  terra::vect(dbs)
  DBI::dbDisconnect(duckdb_conn, shutdown = TRUE)
}
```

`$,dbSpatial-method` *Column extraction for dbSpatial*

Description

Extract a column from a dbSpatial object

Usage

```
## S4 method for signature 'dbSpatial'
x$name
```

Arguments

`x`	A dbSpatial object
`name`	Column name to extract

Value

A vector of values from the specified column

Index

∗ **accessors**
  st_npoints, 17
  st_x, 21
  st_y, 23
∗ **constructors**
  st_as_geojson, 9
  st_as_text, 11
∗ **dbData**
  head,dbSpatial-method, 6
  tail,dbSpatial-method, 25
∗ **dbSpatial**
  as_dbSpatial, 3
  dbSpatial, 4
  show,dbSpatial-method, 7
  st_as_sf.dbSpatial, 10
  vect,dbSpatial-method, 26
∗ **duckdb-ext**
  loadSpatial, 7
∗ **geom_construction**
  st_translate, 20
∗ **geom_scalar**
  st_geometrytype, 14
∗ **geom_summary**
  st_bbox, 12
  st_xmax, 22
  st_ymax, 24
∗ **geometry_ops**
  st_buffer, 12
  st_centroid, 13
  st_simplify, 19
∗ **measurements**
  st_area, 8
  st_length, 16
  st_perimeter, 18
∗ **options**
  dbSpatial_options, 5
$,dbSpatial-method, 27

as_dbSpatial, 3, *5*, *8*, *10*, *26*

dbSpatial, *3*, *4*, 4, *5*, *8–24*, *26*
dbSpatial-class *(*dbSpatial*)*, 4
dbSpatial-options *(*dbSpatial_options*)*, 5
dbSpatial_options, 5

head,dbSpatial-method, 6

loadSpatial, 7

sf::sf, *10*
sf::st_as_sf, *26*
sf::st_buffer(), *13*
sf::st_read, *10*
show,dbSpatial-method, 7
st_area, 8, *16*, *18*
st_as_geojson, 9, *11*
st_as_sf.dbSpatial, *3*, *5*, *8*, 10, *26*
st_as_text, *9*, 11
st_bbox, 12, *22*, *24*
st_buffer, 12, *14*, *19*
st_centroid, *13*, 13, *19*
st_geometrytype, 14
st_join.dbSpatial, 15
st_length, *8*, 16, *18*
st_npoints, 17, *21*, *23*
st_npoints,dbSpatial-method
        *(*st_npoints*)*, 17
st_perimeter, *8*, *16*, 18
st_simplify, *13*, *14*, 19
st_translate, 20
st_translate,dbSpatial-method
        *(*st_translate*)*, 20
st_x, *17*, 21, *23*
st_x,dbSpatial-method *(*st_x*)*, 21
st_xmax, *12*, 22, *24*
st_xmax,dbSpatial-method *(*st_xmax*)*, 22
st_y, *17*, *21*, 23
st_y,dbSpatial-method *(*st_y*)*, 23
st_ymax, *12*, *22*, 24
st_ymax,dbSpatial-method *(*st_ymax*)*, 24

tail,dbSpatial-method, 25
terra::SpatVector, *26*

vect,dbSpatial-method, 26