Package 'tiledb'

May 15, 2025

Type Package

Version 0.32.0

Title Modern Database Engine for Complex Data Based on
Multi-Dimensional Arrays

Description The modern database 'TileDB' introduces a powerful on-disk
format for storing and accessing any complex data based on multi-dimensional
arrays. It supports dense and sparse arrays, dataframes and key-values stores,
cloud storage ('S3', 'GCS', 'Azure'), chunked arrays, multiple compression,
encryption and checksum filters, uses a fully multi-threaded implementation,
supports parallel I/O, data versioning ('time travel'), metadata and groups.
It is implemented as an embeddable cross-platform C++ library with APIs from
several languages, and integrations. This package provides the R support.

Copyright TileDB, Inc.

License MIT + file LICENSE

URL https://github.com/TileDB-Inc/TileDB-R,
https://tiledb-inc.github.io/TileDB-R/

BugReports https://github.com/TileDB-Inc/TileDB-R/issues

SystemRequirements A C++17 compiler is required; on macOS compilation
version 11.0 or later is required. Optionally cmake (only when
TileDB source build selected), curl (only when TileDB source
build selected)), and git (only when TileDB source build
selected); on x86_64 and M1 platforms pre-built TileDB Embedded
libraries are available at GitHub and are used if no TileDB
installation is detected, and no other option to build or
download was specified by the user.

Depends R (>= 4.3)

Imports methods, Rcpp (>= 1.0.8), nanotime, spdl, nanoarrow, tools

LinkingTo Rcpp, RcppInt64, nanoarrow

Suggests tinytest, simplermarkdown, curl, bit64, Matrix,
palmerpenguins, nycflights13, data.table, tibble, arrow

VignetteBuilder simplermarkdown

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author TileDB, Inc. [aut, cph],
        Isaiah Norton [cre]

Maintainer Isaiah Norton <isaiah@tiledb.com>

Repository CRAN

Date/Publication 2025-05-15 18:00:02 UTC

Contents

---

 allows_dups   *Returns logical value whether the array schema allows duplicate val-*
             ues or not. This is only valid for sparse arrays.

Description

Returns logical value whether the array schema allows duplicate values or not. This is only valid
for sparse arrays.

Usage

```
allows_dups(x)

## S4 method for signature 'tiledb_array_schema'
allows_dups(x)

tiledb_array_schema_get_allows_dups(x)
```

Arguments

x   A TileDB Schema object

Value

A logical value.

---

allows_dups<-   *Sets toggle whether the array schema allows duplicate values or not.*
                          This is only valid for sparse arrays.

Description

Sets toggle whether the array schema allows duplicate values or not. This is only valid for sparse arrays.

Usage

```
allows_dups(x) <- value

## S4 replacement method for signature 'tiledb_array_schema'
allows_dups(x) <- value

tiledb_array_schema_set_allows_dups(x, value)
```

Arguments

x   A TileDB Schema object

value   logical value

Value

An object of class `tiledb_array_schema`.

---

array_consolidate *Consolidate fragments of a TileDB Array*

Description

This function invokes a consolidation operation. Parameters affecting the operation can be set via an optional configuration object. Start and end timestamps can also be set directly.

Usage

```
array_consolidate(
  uri,
  cfg = NULL,
  start_time,
  end_time,
  ctx = tiledb_get_context()
)
```

Arguments

uri	A character value with the URI of a TileDB Array
cfg	An optional TileDB Configuration object
start_time	An optional timestamp value, if missing config default is used
end_time	An optional timestamp value, if missing config default is used
ctx	An option TileDB Context object

Value

NULL is returned invisibly

---

array_vacuum *After consolidation, remove consolidated fragments of a TileDB Array*

Description

This function can remove fragments following a consolidation step. Note that vacuuming should *not* be run if one intends to use the TileDB *time-traveling* feature of opening arrays at particular timestamps.

Usage

```
array_vacuum(uri, cfg = NULL, start_time, end_time, ctx = tiledb_get_context())
```

Arguments

`uri`	A character value with the URI of a TileDB Array
`cfg`	An optional TileDB Configuration object
`start_time`	An optional timestamp value, if missing config default is used
`end_time`	An optional timestamp value, if missing config default is used
`ctx`	An option TileDB Context object

Details

Parameters affecting the operation can be set via an optional configuration object. Start and end timestamps can also be set directly.

Value

NULL is returned invisibly

`as.data.frame.tiledb_config`

Convert a `tiledb_config` *object to a R data.frame*

Description

Convert a `tiledb_config` object to a R data.frame

Usage

```
## S3 method for class 'tiledb_config'
as.data.frame(x, ...)
```

Arguments

`x`	`tiledb_config` object
`...`	Extra parameter for method signature, currently unused.

Value

a data.frame wth parameter, value columns

Examples

```
cfg <- tiledb_config()
as.data.frame(cfg)
```

---

as.vector.tiledb_config

Convert a tiledb_config *object to a R vector*

Description

Convert a tiledb_config object to a R vector

Usage

```
## S3 method for class 'tiledb_config'
as.vector(x, mode = "any")
```

Arguments

x	tiledb_config object
mode	Character value "any", currently unused

Value

a character vector of config parameter names, values

Examples

```
cfg <- tiledb_config()
as.vector(cfg)
```

---

attrs,tiledb_array,ANY-method

Retrieve attributes from tiledb_array *object*

Description

By default, all attributes will be selected. But if a subset of attribute names is assigned to the internal slot attrs, then only those attributes#' will be queried. This methods accesses the slot.

Usage

```
## S4 method for signature 'tiledb_array,ANY'
attrs(object)
```

Arguments

object	A tiledb_array object

Value

An empty character vector if no attributes have been selected or else a vector with attributes; NA means no attributes will be returned.

---

attrs,tiledb_array_schema,ANY-method

> *Returns a list of all* tiledb_attr *objects associated with the* tiledb_array_schema

---

Description

Returns a list of all tiledb_attr objects associated with the tiledb_array_schema

Usage

```
## S4 method for signature 'tiledb_array_schema,ANY'
attrs(object, idx, ...)
```

Arguments

object	A TileDB Schema object
idx	index argument, currently unused.
...	Extra parameter for method signature, currently unused.

Value

A list of tiledb_attr objects

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(
  tiledb_attr("a1", type = "INT32"),
  tiledb_attr("a2", type = "FLOAT64")
))
attrs(sch)

lapply(attrs(sch), datatype)
```

---

attrs,tiledb_array_schema,character-method

> *Returns a* tiledb_attr *object associated with the* tiledb_array_schema *with a given name.*

Description

Returns a tiledb_attr object associated with the tiledb_array_schema with a given name.

Usage

```
## S4 method for signature 'tiledb_array_schema,character'
attrs(object, idx, ...)
```

Arguments

object	A TileDB Schema object
idx	attribute name string
...	Extra parameter for method signature, currently unused.

Value

A tiledb_attr object

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(
  tiledb_attr("a1", type = "INT32"),
  tiledb_attr("a2", type = "FLOAT64")
))
attrs(sch, "a2")
```

---

attrs,tiledb_array_schema,numeric-method

> *Returns a* tiledb_attr *object associated with the* tiledb_array_schema *with a given index*

Description

The attribute index is defined by the order the attributes were defined in the schema

Usage

```
## S4 method for signature 'tiledb_array_schema,numeric'
attrs(object, idx, ...)
```

Arguments

object	A TileDB Schema object
idx	attribute index
...	Extra parameter for method signature, currently unused.

Value

A `tiledb_attr` object

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(
  tiledb_attr("a1", type = "INT32"),
  tiledb_attr("a2", type = "FLOAT64")
))
attrs(sch, 2)
```

---

attrs<-,tiledb_array-method
                        Selects attributes for the given TileDB array

---

Description

Selects attributes for the given TileDB array

Usage

```
## S4 replacement method for signature 'tiledb_array'
attrs(x) <- value
```

Arguments

| x | A `tiledb_array` object |
| value | A character vector with attributes; the value `NA_character_` signals no attributes should be returned; default is an empty character vector implying all columns are returned. |

Value

The modified `tiledb_array` object

---

capacity   *Retrieve schema capacity (for sparse fragments)*

Description

Returns the `tiledb_array` schema tile capacity for sparse fragments.

Usage

```
capacity(object)

## S4 method for signature 'tiledb_array_schema'
capacity(object)

tiledb_array_schema_get_capacity(object)
```

Arguments

object   A TileDB Schema object

Value

The tile capacity value

---

capacity<-   *Sets the schema capacity (for sparse fragments)*

Description

Sets the `tiledb_array` schema tile capacity for sparse fragments.

Usage

```
capacity(x) <- value

## S4 replacement method for signature 'tiledb_array_schema'
capacity(x) <- value

tiledb_array_schema_set_capacity(x, value)
```

Arguments

x   A TileDB Schema object

value   An integer or numeric value for the new tile capacity

Value

The modified `tiledb_array_schema` object.

`cell_order,tiledb_array_schema-method`

Returns the cell layout string associated with the `tiledb_array_schema`

Description

Returns the cell layout string associated with the `tiledb_array_schema`

Usage

```
## S4 method for signature 'tiledb_array_schema'
cell_order(object)
```

Arguments

object   A TileDB Schema object

Value

A character string with cell's layout, e.g., `"COL_MAJOR"`.

`cell_val_num`   *Return the number of scalar values per attribute cell*

Description

Return the number of scalar values per attribute cell

Usage

```
cell_val_num(object)

## S4 method for signature 'tiledb_attr'
cell_val_num(object)

tiledb_attribute_get_cell_val_num(object)
```

Arguments

object   `tiledb_attr` object

Value

integer number of cells

Examples

```
a1 <- tiledb_attr("a1", type = "FLOAT64", ncells = 1)
cell_val_num(a1)
```

---

cell_val_num,tiledb_dim-method
Return the number of scalar values per dimension cell

Description

Return the number of scalar values per dimension cell

Usage

```
## S4 method for signature 'tiledb_dim'
cell_val_num(object)

tiledb_dim_get_cell_val_num(object)
```

Arguments

object   A `tiledb_dim` object

Value

integer number of cells

---

cell_val_num<-   *Set the number of scalar values per attribute cell*

Description

Set the number of scalar values per attribute cell

Usage

```
cell_val_num(x) <- value

## S4 replacement method for signature 'tiledb_attr'
cell_val_num(x) <- value

tiledb_attribute_set_cell_val_num(x, value)
```

Arguments

x	A TileDB Attribute object
value	An integer value of number of cells

Value

The modified attribute is returned

---

completedBatched   *Check 'batched' query for completion*

Description

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

Usage

```
completedBatched(obj)
```

Arguments

obj	A list object as returned by createBatched

Value

A logical value to indicated if the query completed

---

config,tiledb_ctx-method

                        Retrieve the tiledb_config *object from the* tiledb_ctx

---

Description

Retrieve the tiledb_config object from the tiledb_ctx

Usage

```
## S4 method for signature 'tiledb_ctx'
config(object = tiledb_get_context())
```

Arguments

object	tiledb_ctx object

Value

tiledb_config object associated with the tiledb_ctx instance

Examples

```
ctx <- tiledb_ctx(c("sm.tile_cache_size" = "10"))
cfg <- config(ctx)
cfg["sm.tile_cache_size"]
```

---

createBatched   *Create a 'batched' query object*

Description

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

Usage

```
createBatched(x)
```

Arguments

x   A tiledb_array object

Details

The tiledb_array object can be parameterised as usual.

Value

A batchedquery object, that is a list containing an external pointer to a TileDB Query object along with other support variables used by fetchBatched

datatype,tiledb_attr-method

Return the tiledb_attr *datatype*

Description

Return the tiledb_attr datatype

Usage

```
## S4 method for signature 'tiledb_attr'
datatype(object)
```

Arguments

object   tiledb_attr object

Value

tiledb datatype string

Examples

```
a1 <- tiledb_attr("a1", type = "INT32")
datatype(a1)

a2 <- tiledb_attr("a1", type = "FLOAT64")
datatype(a2)
```

datatype,tiledb_dim-method

Return the tiledb_dim *datatype*

Description

Return the tiledb_dim datatype

Usage

```
## S4 method for signature 'tiledb_dim'
datatype(object)
```

Arguments

object   A tiledb_dim object

Value

A character string with tiledb's datatype.

Examples

```
d1 <- tiledb_dim("d1", domain = c(5L, 10L), tile = 2L, type = "INT32")
datatype(d1)
```

---

datatype,tiledb_domain-method

Returns the tiledb_domain TileDB type string

Description

Returns the tiledb_domain TileDB type string

Usage

```
## S4 method for signature 'tiledb_domain'
datatype(object)
```

Arguments

object   tiledb_domain

Value

tiledb_domain type string

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32")))
datatype(dom)
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64")))
datatype(dom)
```

---

datetimes_as_int64 *Retrieve datetimes_as_int64 toggle*

Description

A `tiledb_array` object may contain date and datetime objects. While their internal representation is generally shielded from the user, it can useful to access them as the 'native' format which is an `integer64`. This function retrieves the current value of the selection variable, which has a default of `FALSE`.

Usage

```
datetimes_as_int64(object)

## S4 method for signature 'tiledb_array'
datetimes_as_int64(object)
```

Arguments

object   A `tiledb_array` object

Value

A logical value indicating whether `datetimes_as_int64` is selected

---

datetimes_as_int64<- *Set datetimes_as_int64 toggle*

Description

A `tiledb_array` object may contain date and datetime objects. While their internal representation is generally shielded from the user, it can useful to access them as the 'native' format which is an `integer64`. This function set the current value of the selection variable, which has a default of `FALSE`.

Usage

```
datetimes_as_int64(x) <- value

## S4 replacement method for signature 'tiledb_array'
datetimes_as_int64(x) <- value
```

Arguments

x   A `tiledb_array` object
value   A logical value with the selection

Value

The modified `tiledb_array` array object

---

describe	*Describe a TileDB array schema via code to create it*

Description

Note that this function is an unexported internal function that can be called using the colons as in `tiledb:::describe(arr)`.

Usage

```
describe(arr)
```

Arguments

arr   A TileDB Array object

Value

Nothing is returned as the function is invoked for the side effect of printing the schema via a sequence of R instructions to re-create it.

---

dim.tiledb_array_schema	
	Retrieve the dimension (domain extent) of the domain

Description

Only valid for integral (integer) domains

Usage

```
## S3 method for class 'tiledb_array_schema'
dim(x)
```

Arguments

x   A TileDB Schema object

Value

a dimension vector

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(
  tiledb_attr("a1", type = "INT32"),
  tiledb_attr("a2", type = "FLOAT64")
))
dim(sch)
```

---

dim.tiledb_dim   *Retrieves the dimension of the tiledb_dim domain*

Description

Retrieves the dimension of the tiledb_dim domain

Usage

```
## S3 method for class 'tiledb_dim'
dim(x)
```

Arguments

x   A `tiledb_dim` object

Value

A vector of the tile_dim domain type, of the dim domain dimension (extent)

Examples

```
d1 <- tiledb_dim("d1", c(1L, 10L), 5L)
dim(d1)
```

---

dim.tiledb_domain   *Retrieve the dimension (domain extent) of the domain*

Description

Only valid for integral (integer) domains

Usage

```
## S3 method for class 'tiledb_domain'
dim(x)
```

Arguments

x   tiledb_domain

Value

dimension vector

Examples

```
dom <- tiledb_domain(dims = c(
  tiledb_dim("d1", c(1L, 100L), type = "INT32"),
  tiledb_dim("d2", c(1L, 100L), type = "INT32")
))
dim(dom)
```

```
dimensions,tiledb_array_schema-method
                        Returns a list of tiledb_dim objects associated with the
                        tiledb_array_schema
```

Description

Returns a list of tiledb_dim objects associated with the tiledb_array_schema

Usage

```
## S4 method for signature 'tiledb_array_schema'
dimensions(object)
```

Arguments

object   A TileDB Schema object

Value

A list of tiledb_dim objects

Examples

```
dom <- tiledb_domain(dims = c(
  tiledb_dim("d1", c(1L, 100L), type = "INT32"),
  tiledb_dim("d2", c(1L, 50L), type = "INT32")
))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32")))
dimensions(dom)

lapply(dimensions(dom), name)
```

`dimensions,tiledb_domain-method`
                    Returns a list of the tiledb_domain dimension objects

---

Description

Returns a list of the tiledb_domain dimension objects

Usage

```
## S4 method for signature 'tiledb_domain'
dimensions(object)
```

Arguments

object   tiledb_domain

Value

a list of tiledb_dim

Examples

```
dom <- tiledb_domain(dims = c(
  tiledb_dim("d1", c(1L, 100L), type = "INT32"),
  tiledb_dim("d2", c(1L, 50L), type = "INT32")
))
dimensions(dom)

lapply(dimensions(dom), name)
```

`domain,tiledb_array_schema-method`
                    Returns the `tiledb_domain` *object associated with a given*
                    `tiledb_array_schema`

Description

Returns the `tiledb_domain` object associated with a given `tiledb_array_schema`

Usage

```
## S4 method for signature 'tiledb_array_schema'
domain(object)
```

Arguments

object   A TileDB Schema object

Value

A `tiledb_domain` object

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32")))
domain(sch)
```

---

domain,tiledb_dim-method

Return the `tiledb_dim` *domain*

Description

Return the `tiledb_dim` domain

Usage

```
## S4 method for signature 'tiledb_dim'
domain(object)
```

Arguments

object   A `tiledb_dim` object

Value

a vector of (lb, ub) inclusive domain of the dimension

Examples

```
d1 <- tiledb_dim("d1", domain = c(5L, 10L))
domain(d1)
```

extended   *Retrieve data.frame extended returns columns toggle*

Description

A `tiledb_array` object can be returned as `data.frame`. This methods returns the selection value for 'extended' format including row (and column, if present) indices.

Usage

```
extended(object)

## S4 method for signature 'tiledb_array'
extended(object)
```

Arguments

object   A `tiledb_array` object

Value

A logical value indicating whether an `extended` return is selected

extended<-   *Set data.frame extended return columns toggle*

Description

A `tiledb_array` object can be returned as `data.frame`. This methods set the selection value for 'extended' format including row (and column, if present) indices.

Usage

```
extended(x) <- value

## S4 replacement method for signature 'tiledb_array'
extended(x) <- value
```

Arguments

x   A `tiledb_array` object
value   A logical value with the selection

Value

The modified `tiledb_array` array object

---

fetchBatched *Run a 'batched' query*

Description

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

Usage

```
fetchBatched(x, obj)
```

Arguments

x	A `tiledb_array` object
obj	A `batchedquery` object as returned by `createBatched`

Details

The `tiledb_array` object can be parameterised as usual.

Value

A data.frame object with the (potentially partial) result of a batched query

---

filter_list,tiledb_array_schema-method

Returns the offsets and coordinate filter_lists associated with the `tiledb_array_schema`

Description

Returns the offsets and coordinate filter_lists associated with the `tiledb_array_schema`

Usage

```
## S4 method for signature 'tiledb_array_schema'
filter_list(object)
```

Arguments

object	A TileDB Schema object

Value

A list of `tiledb_filter_list` objects

`filter_list,tiledb_attr-method`

Returns the TileDB Filter List object associated with the given TileDB Attribute

---

Description

Returns the TileDB Filter List object associated with the given TileDB Attribute

Usage

```
## S4 method for signature 'tiledb_attr'
filter_list(object)
```

Arguments

object   TileDB Attribute

Value

a tiledb_filter_list object

Examples

```
attr <- tiledb_attr(
  type = "INT32",
  filter_list = tiledb_filter_list(list(tiledb_filter("ZSTD")))
)
filter_list(attr)
```

`filter_list,tiledb_dim-method`

Returns the TileDB Filter List object associated with the given TileDB Dimension

---

Description

Returns the TileDB Filter List object associated with the given TileDB Dimension

Usage

```
## S4 method for signature 'tiledb_dim'
filter_list(object)
```

Arguments

object   A `tiledb_dim` object

Value

A `tiledb_filter_list` object

`filter_list<-,tiledb_attr-method`
Sets the TileDB Filter List for the TileDB Attribute object

Description

Sets the TileDB Filter List for the TileDB Attribute object

Usage

```
## S4 replacement method for signature 'tiledb_attr'
filter_list(x) <- value
```

Arguments

x   TileDB Attribute

value   TileDB Filter List

Value

The modified TileDB Attribute object

`filter_list<-,tiledb_dim-method`
Sets the TileDB Filter List for the TileDB Dimension object

Description

Sets the TileDB Filter List for the TileDB Dimension object

Usage

```
## S4 replacement method for signature 'tiledb_dim'
filter_list(x) <- value
```

Arguments

x   A `tiledb_dim` object

value   TileDB Filter List

Value

The modified TileDB Dimension object

---

fromDataFrame	*Create a TileDB dense or sparse array from a given* data.frame *Object*

Description

The supplied data.frame object is (currently) limited to integer, numeric, or character. In addition, three datetime columns are supported with the R representations of Date, POSIXct and nanotime.

Usage

```
fromDataFrame(
  obj,
  uri,
  col_index = NULL,
  sparse = TRUE,
  allows_dups = sparse,
  cell_order = "COL_MAJOR",
  tile_order = "COL_MAJOR",
  filter = "ZSTD",
  capacity = 10000L,
  tile_domain = NULL,
  tile_extent = NULL,
  mode = c("ingest", "schema_only", "append"),
  filter_list = NULL,
  coords_filters = "ZSTD",
  offsets_filters = "ZSTD",
  validity_filters = "RLE",
  debug = FALSE,
  timestamps = as.POSIXct(double(), origin = "1970-01-01")
)
```

Arguments

obj	A data.frame object.
uri	A character variable with an Array URI.
col_index	An optional column index, either numeric with a column index, or character with a column name, designating an index column; default is NULL implying an index column is added when the array is created
sparse	A logical switch to select sparse (the default) or dense
allows_dups	A logical switch to select if duplicate values are allowed or not, default is the same value as 'sparse'.

cell_order	A character variable with one of the TileDB cell order values, default is "COL_MAJOR".
tile_order	A character variable with one of the TileDB tile order values, default is "COL_MAJOR".
filter	A character variable vector, defaults to 'ZSTD', for one or more filters to be applied to each attribute;
capacity	A integer value with the schema capacity, default is 10000.
tile_domain	An integer vector or list or NULL. If an integer vector of size two it specifies the integer domain of the row dimension; if a list then a named element is used for the dimension of the same name; or if NULL the row dimension of the obj is used.
tile_extent	An integer value for the tile extent of the row dimensions; if NULL the row dimension of the obj is used. Note that the tile_extent cannot exceed the tile domain.
mode	A character variable with possible values 'ingest' (for schema creation and data ingestion, the default behavior), 'schema_only' (to create the array schema without writing to the newly-created array) and 'append' (to only append to an already existing array).
filter_list	A named list specifying filter choices per column, default is an empty list object. This argument applies for all named arguments and the matchin dimensions or attributes. The filter argument still applies for all unnamed arguments.
coords_filters	A character vector with filters for coordinates, default is ZSTD.
offsets_filters	
	A character vector with filters for coordinates, default is ZSTD.
validity_filters	
	A character vector with filters for coordinates, default is RLE.
debug	Logical flag to select additional output.
timestamps	Vector with up to two POSIXct variables denoting open intervals; default is length zero where start and end are set (implicitly) to current time; in case of one value it is used as the interval end, and in case of two values they are taken as start and end. This applies to write and append modes only and not to schema creation.

Details

The created (dense or sparse) array will have as many attributes as there are columns in the data.frame. Each attribute will be a single column. For a sparse array, one or more columns have to be designated as dimensions.

At present, factor variable are converted to character.

Value

Null, invisibly.

Examples

```
uri <- tempfile()
fromDataFrame(iris, uri)
arr <- tiledb_array(uri, return_as="data.frame", extended=FALSE)
newdf <- arr[]
all.equal(iris, newdf, check.attributes=FALSE)  # extra attribute on query in newdf
all.equal(as.matrix(iris), as.matrix(newdf)) # also strips attribute
```

---

fromMatrix   *Create a TileDB array from an R matrix, or return an R matrix*

Description

The functions `fromMatrix` and `toMatrix` help in storing (and retrieving) matrices using a TileDB backend. In particular they help for matrices with explicit rownames.

Usage

```
fromMatrix(obj, uri, filter = "ZSTD", capacity = 10000L)

toMatrix(uri)
```

Arguments

obj	A sparse matrix object.
uri	A character variable with an Array URI.
filter	A character variable vector, defaults to 'ZSTD', for one or more filters to be applied to each attribute;
capacity	A integer value with the schema capacity, default is 10000.

Value

Null, invisibly.

---

fromSparseMatrix   *Create (or return) a TileDB sparse array*

Description

The functions `fromSparseMatrix` and `toSparseMatrix` help in storing (and retrieving) sparse matrices using a TileDB backend.

Usage

```
fromSparseMatrix(
  obj,
  uri,
  cell_order = "ROW_MAJOR",
  tile_order = "ROW_MAJOR",
  filter = "ZSTD",
  capacity = 10000L
)

toSparseMatrix(uri)
```

Arguments

`obj`	A sparse matrix object.
`uri`	A character variable with an Array URI.
`cell_order`	A character variable with one of the TileDB cell order values, default is "COL_MAJOR".
`tile_order`	A character variable with one of the TileDB tile order values, default is "COL_MAJOR".
`filter`	A character variable vector, defaults to 'ZSTD', for one or more filters to be applied to each attribute;
`capacity`	A integer value with the schema capacity, default is 10000.

Value

Null, invisibly.

Examples

```
## Not run:
if (requireNamespace("Matrix", quietly=TRUE)) {
  library(Matrix)
  set.seed(123)      # just to fix it
  mat <- matrix(0, nrow=20, ncol=10)
  mat[sample(seq_len(200), 20)] <- seq(1, 20)
  spmat <- as(mat, "dgTMatrix")  # sparse matrix in dgTMatrix format
  uri <- "sparse_matrix"
  fromSparseMatrix(spmat, uri)   # now written
  chk <- toSparseMatrix(uri)     # and re-read
  print(chk)
  all.equal(spmat, chk)
}

## End(Not run)
```

---

generics   *Generic Methods*

Description

Definition of generic methods

Usage

```
schema(object, ...)

return.data.frame(object, ...)

return.data.frame(x) <- value

attrs(x) <- value

raw_dump(object, ...)

domain(object, ...)

dimensions(object, ...)

attrs(object, idx, ...)

cell_order(object, ...)

tile_order(object, ...)

filter_list(object, ...)

filter_list(x) <- value

is.sparse(object, ...)

tiledb_ndim(object, ...)

name(object)

datatype(object)

config(object, ...)

tile(object)

is.integral(object)
```

```
nfilters(object)

tdb_filter(x, ...)

tdb_select(x, ...)

tdb_collect(x, ...)
```

Arguments

object	A TileDB object
...	Currently unused
x	A TileDB Object
value	A value to be assigned
idx	An index argument

---

has_attribute	*Check a schema for a given attribute name*

Description

Check a schema for a given attribute name

Usage

```
has_attribute(schema, attr)
```

Arguments

schema	A TileDB Schema object
attr	A character variable with an attribute name

Value

A boolean value indicating if the attribute exists in the schema

---

is.anonymous   *Returns TRUE if the tiledb_dim is anonymous*

Description

A TileDB attribute is anonymous if no name/label is defined

Usage

```
is.anonymous(object)

## S3 method for class 'tiledb_attr'
is.anonymous(object)
```

Arguments

object   tiledb_attr object

Value

TRUE or FALSE

Examples

```
a1 <- tiledb_attr("a1", type = "FLOAT64")
is.anonymous(a1)

a2 <- tiledb_attr("", type = "FLOAT64")
is.anonymous(a2)
```

---

is.anonymous.tiledb_dim

Returns TRUE if the tiledb_dim is anonymous

Description

A TileDB dimension is anonymous if no name/label is defined

Usage

```
## S3 method for class 'tiledb_dim'
is.anonymous(object)
```

Arguments

object   A tiledb_dim object

Value

TRUE or FALSE

Examples

```
d1 <- tiledb_dim("d1", c(1L, 10L), 10L)
is.anonymous(d1)

d2 <- tiledb_dim("", c(1L, 10L), 10L)
is.anonymous(d2)
```

```
is.integral,tiledb_domain-method
```
Returns TRUE is tiledb_domain is an integral (integer) domain

---

Description

Returns TRUE is tiledb_domain is an integral (integer) domain

Usage

```
## S4 method for signature 'tiledb_domain'
is.integral(object)
```

Arguments

object   tiledb_domain

Value

TRUE if the domain is an integral domain, else FALSE

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32")))
is.integral(dom)
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64")))
is.integral(dom)
```

is.sparse,tiledb_array_schema-method
Returns TRUE if the tiledb_array_schema is sparse, else FALSE

Description

Returns TRUE if the tiledb_array_schema is sparse, else FALSE

Usage

```
## S4 method for signature 'tiledb_array_schema'
is.sparse(object)
```

Arguments

object	A TileDB Schema object

Value

TRUE if tiledb_array_schema object is sparse

limitTileDBCores   *Limit TileDB core use to a given number of cores*

Description

By default, TileDB will use all available cores on a given machine. In multi-user or multi-process settings, one may want to reduce the number of core. This function will take a given number, or default to smaller of the 'Ncpus' options value or the '"OMP_THREAD_LIMIT"' enviroment variable (or two as hard fallback).

Usage

```
limitTileDBCores(ncores, verbose = FALSE)
```

Arguments

ncores	Value of CPUs used, if missing the smaller of a fallback of two, the value of 'Ncpus' (if set) and the value of environment variable '"OMP_THREAD_LIMIT"' is used.
verbose	Optional logical toggle; if set, a short message is displayed informing the user about the value set.

Details

As this function returns a config object, its intended use is as argument to the context creating functions: `ctx <- tiledb_ctx(limitTileDBCores())`. To check that the values are set (or at a later point, still set) the config object should be retrieved via the corresponding method and this `ctx` object: `cfg <- config(ctx)`.

Value

The modified configuration object is returned invisibly.

---

max_chunk_size   *Returns the filter_list's max_chunk_size*

Description

Returns the filter_list's max_chunk_size

Usage

```
max_chunk_size(object)

## S4 method for signature 'tiledb_filter_list'
max_chunk_size(object)

tiledb_filter_list_get_max_chunk_size(object)
```

Arguments

object   tiledb_filter_list

Value

integer max_chunk_size

Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
max_chunk_size(filter_list)
```

---

name,tiledb_attr-method

Return the tiledb_attr *name*

Description

Return the tiledb_attr name

Usage

```
## S4 method for signature 'tiledb_attr'
name(object)
```

Arguments

object   tiledb_attr object

Value

string name, empty string if the attribute is anonymous

Examples

```
a1 <- tiledb_attr("a1", type = "INT32")
name(a1)

a2 <- tiledb_attr(type = "INT32")
name(a2)
```

---

name,tiledb_dim-method

Return the tiledb_dim *name*

Description

Return the tiledb_dim name

Usage

```
## S4 method for signature 'tiledb_dim'
name(object)
```

Arguments

object   A tiledb_dim object

Value

string name, empty string if the dimension is anonymous

Examples

```
d1 <- tiledb_dim("d1", c(1L, 10L))
name(d1)

d2 <- tiledb_dim("", c(1L, 10L))
name(d2)
```

---

nfilters,tiledb_filter_list-method

Returns the filter_list's number of filters

Description

Returns the filter_list's number of filters

Usage

```
## S4 method for signature 'tiledb_filter_list'
nfilters(object)
```

Arguments

object   tiledb_filter_list

Value

integer number of filters

Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
nfilters(filter_list)
```

parse_query_condition   *Create a 'tiledb_query_condition' object from an expression*

Description

The grammar for query conditions is at present constraint to eight operators (">", ">=", "<", "<=", "==", "!=", "%in%", "%nin%"), and three boolean operators ("&&", also as "&", ("||", also as "|", and "!" for negation. Note that we locally define "%nin%" as Negate() call around %in%) which extends R a little for this use case.

Usage

```
parse_query_condition(
  expr,
  ta = NULL,
  debug = FALSE,
  strict = TRUE,
  use_int64 = FALSE
)
```

Arguments

expr	An expression that is understood by the TileDB grammar for query conditions.
ta	A tiledb_array object that the query condition is applied to; this argument is optional in some cases but required in some others.
debug	A boolean toogle to enable more verbose operations, defaults to 'FALSE'.
strict	A boolean toogle to, if set, errors if a non-existing attribute is selected or filtered on, defaults to 'TRUE'; if 'FALSE' a warning is shown by execution proceeds.
use_int64	A boolean toggle to switch to integer64 if integer is seen, default is false to remain as a default four-byte int

Details

Expressions are parsed locally by this function. The debug=TRUE option may help if an issue has to be diagnosed. In most cases of an errroneous parse, it generally helps to supply the tiledb_array providing schema information. One example are numeric and integer columns where the data type is difficult to guess. Also, when using the "%in%" or "%nin%" operators, the argument is mandatory.

Value

A tiledb_query_condition object

Examples

```
## Not run:
uri <- "mem://airquality" # change to on-disk for persistence
fromDataFrame(airquality, uri, col_index = c("Month", "Day")) # dense array
## query condition on dense array requires extended=FALSE
tiledb_array(uri,
  return_as = "data.frame", extended = FALSE,
  query_condition = parse_query_condition(Temp > 90)
)[]

## End(Not run)
```

---

print.tiledb_metadata   *Print a TileDB Array Metadata object*

Description

Print a TileDB Array Metadata object

Usage

```
## S3 method for class 'tiledb_metadata'
print(x, width = NULL, ...)
```

Arguments

x	A TileDB array object
width	Optional display width, defaults to NULL
...	Optional method arguments, currently unused

Value

The array object, invisibly

---

query_condition   *Retrieve query_condition value for the array*

Description

A `tiledb_array` object can have a corresponding query condition object. This methods returns it.

Usage

```
query_condition(object)

## S4 method for signature 'tiledb_array'
query_condition(object)
```

Arguments

object   A `tiledb_array` object

Value

A `tiledb_query_condition` object

---

query_condition<-   *Set query_condition object for the array*

Description

A `tiledb_array` object can have an associated query condition object to set conditions on the read queries. This methods sets the 'query_condition' object.

Usage

```
query_condition(x) <- value

## S4 replacement method for signature 'tiledb_array'
query_condition(x) <- value
```

Arguments

x   A `tiledb_array` object

value   A `tiledb_query_conditon_object`

Value

The modified `tiledb_array` array object

---

query_layout   *Retrieve query_layout values for the array*

Description

A `tiledb_array` object can have a corresponding query with a given layout given layout. This methods returns the selection value for 'query_layout' as a character value.

Usage

```
query_layout(object)

## S4 method for signature 'tiledb_array'
query_layout(object)
```

Arguments

object	A `tiledb_array` object

Value

A character value describing the query layout

---

query_layout<- *Set query_layout return values for the array*

Description

A `tiledb_array` object can have an associated query with a specific layout. This methods sets the selection value for 'query_layout' from a character value.

Usage

```
query_layout(x) <- value

## S4 replacement method for signature 'tiledb_array'
query_layout(x) <- value
```

Arguments

x	A `tiledb_array` object
value	A character variable for the query layout. Permitted values are "ROW_MAJOR", "COL_MAJOR", "GLOBAL_ORDER", or "UNORDERD".

Value

The modified `tiledb_array` array object

---

query_statistics *Retrieve query_statistics toggle*

Description

A `tiledb_array` object can, if requested, return query statistics as a JSON string in an attribute 'query_statistics' attached to the return object. The default value of the logical switch is 'FALSE'. This method returns the current value.

Usage

```
query_statistics(object, ...)

## S4 method for signature 'tiledb_array'
query_statistics(object)
```

Arguments

object	A `tiledb_array` object
...	Currently unused

Value

A logical value indicating whether query statistics are returned.

  `query_statistics<-`   *Set query_statistics toggle*

Description

A `tiledb_array` object can, if requested, return query statistics as a JSON string in an attribute 'query_statistics' attached to the return object. The default value of the logical switch is 'FALSE'. This method sets the value.

Usage

```
query_statistics(x) <- value

## S4 replacement method for signature 'tiledb_array'
query_statistics(x) <- value
```

Arguments

x	A `tiledb_array` object
value	A logical value with the selection

Value

The modified `tiledb_array` array object

  `raw_dump,tiledb_array_schema-method`
                    Raw display of an array schema object

Description

This method used the display method provided by the underlying library.

Usage

```
## S4 method for signature 'tiledb_array_schema'
raw_dump(object)
```

Arguments

object   A TileDB Schema object

`raw_dump,tiledb_attr-method`
Raw display of an attribute object

Description

This method used the display method provided by the underlying library.

Usage

```
## S4 method for signature 'tiledb_attr'
raw_dump(object)
```

Arguments

object   An attribute object

`raw_dump,tiledb_domain-method`
Raw display of a domain object

Description

This method used the display method provided by the underlying library.

Usage

```
## S4 method for signature 'tiledb_domain'
raw_dump(object)
```

Arguments

object   A domain object

---

return.array   *Retrieve array return toggle*

Description

A `tiledb_array` object can be returned as an array (or list of arrays), or, if select, as a `data.frame` or as a `matrix`. This methods returns the selection value for the `array` selection.

Usage

```
return.array(object, ...)

## S4 method for signature 'tiledb_array'
return.array(object)
```

Arguments

object	A `tiledb_array` object
...	Currently unused

Value

A logical value indicating whether `array` return is selected

---

return.array<-   *Set array return toggle*

Description

A `tiledb_array` object can be returned as an array (or list of arrays), or, if select, as a `data.frame` or a `matrix`. This methods sets the selection value for a `array`.

Usage

```
return.array(x) <- value

## S4 replacement method for signature 'tiledb_array'
return.array(x) <- value
```

Arguments

x	A `tiledb_array` object
value	A logical value with the selection

Value

The modified `tiledb_array` array object

```
return.data.frame,tiledb_array-method
```
Retrieve data.frame return toggle

Description

A `tiledb_array` object can be returned as an array (or list of arrays), or, if select, as a `data.frame`. This methods returns the selection value.

Usage

```
## S4 method for signature 'tiledb_array'
return.data.frame(object)
```

Arguments

object   A `tiledb_array` object

Value

A logical value indicating whether `data.frame` return is selected

```
return.data.frame<-,tiledb_array-method
```
Set data.frame return toggle

Description

A `tiledb_array` object can be returned as an array (or list of arrays), or, if select, as a `data.frame`. This methods sets the selection value.

Usage

```
## S4 replacement method for signature 'tiledb_array'
return.data.frame(x) <- value
```

Arguments

x   A `tiledb_array` object

value   A logical value with the selection

Value

The modified `tiledb_array` array object

---

return.matrix   *Retrieve matrix return toggle*

Description

A `tiledb_array` object can be returned as an array (or list of arrays), or, if select, as a `data.frame` or as a `matrix`. This methods returns the selection value for the `matrix` selection.

Usage

```
return.matrix(object, ...)

## S4 method for signature 'tiledb_array'
return.matrix(object)
```

Arguments

object	A `tiledb_array` object
...	Currently unused

Value

A logical value indicating whether `matrix` return is selected

---

return.matrix<-   *Set matrix return toggle*

Description

A `tiledb_array` object can be returned as an array (or list of arrays), or, if select, as a `data.frame` or a `matrix`. This methods sets the selection value for a `matrix`.

Usage

```
return.matrix(x) <- value

## S4 replacement method for signature 'tiledb_array'
return.matrix(x) <- value
```

Arguments

x	A `tiledb_array` object
value	A logical value with the selection

Value

The modified `tiledb_array` array object

---

return_as	*Retrieve return_as conversion preference*

Description

A `tiledb_array` object can be returned as a 'list' (default), 'array', 'matrix', 'data.frame', 'data.table' or 'tibble'. This method permits to select a preference for the returned object. The default value of 'asis' means that no conversion is performed.

Usage

```
return_as(object, ...)

## S4 method for signature 'tiledb_array'
return_as(object)
```

Arguments

object	A `tiledb_array` object
...	Currently unused

Value

A character value indicating the preferred conversion where the value is one of 'asis' (the default), 'array', 'matrix','data.frame', 'data.table', or 'tibble'.

---

return_as<-	*Retrieve return_as conversion preference*

Description

A `tiledb_array` object can be returned as a 'list' (default), 'array', 'matrix', 'data.frame', 'data.table' or 'tibble'. This method This methods permits to set a preference of returning a `list`, `array`, `matrix`, `data.frame`, a `data.table`, or a `tibble`. The default value of "asis" means that no conversion is performed and a `list` is returned.

Usage

```
return_as(x) <- value

## S4 replacement method for signature 'tiledb_array'
return_as(x) <- value
```

Arguments

x   A `tiledb_array` object

value   A character value with the selection

Value

The modified `tiledb_array` array object

---

r_to_tiledb_type   *Look up TileDB type corresponding to the type of an R object*

Description

Look up TileDB type corresponding to the type of an R object

Usage

```
r_to_tiledb_type(x)
```

Arguments

x   an R array or list

Value

single character, e.g. INT32

---

save_allocation_size_preference
                        Store allocation size preference

Description

Save (or load) allocation size default preference in an optional config file

Usage

```
save_allocation_size_preference(value)

load_allocation_size_preference()

get_allocation_size_preference()

set_allocation_size_preference(value)
```

Arguments

value   A numeric value with the desired allocation size (in bytes).

Details

When retrieving data from sparse arrays, allocation sizes cannot be determined *ex ante* as the degree of sparsity is unknown. A configuration value can aide in providing an allocation size value. These functions let the user store such a value for retrieval by their package or script code. The preference will be encoded in a configuration file as R (version 4.0.0 or later) allows a user- and package specific configuration files. These helper functions sets and retrieve the value, respectively, or retrieve the cached value from the package environment where is it set at package load.

The value will be stored as a character value and reparsed so '1e6' and '1000000' are equivalent, and the fixed (but adjustable) number of digits for numerical precision *use for formatting* will impact the writing. This should have no effect on standard allocation sizes.

The value is used as a limit *per column* so total memory use per query will a multiple of this value, and increasing in dimension and attribute count.

A fallback value of 10 mb is used if no user value is set.

Value

For the setter, TRUE is returned invisibly but the function is invoked for the side effect of storing the value. For the getters, the value as a numeric.

Note

This function requires R version 4.0.0 or later to utilise the per-user config directory accessor function. For older R versions, a fallback from the TileDB configuration object is used.

---

save_return_as_preference

Store object conversion preference

Description

Save (or load) 'return_as' conversion preference in an optional config file

Usage

```
save_return_as_preference(
  value = c("asis", "array", "matrix", "data.frame", "data.table", "tibble")
)

load_return_as_preference()

get_return_as_preference()
```

```
set_return_as_preference(
  value = c("asis", "array", "matrix", "data.frame", "data.table", "tibble")
)
```

Arguments

value   A character variable with one of the six permitted values

Details

The `tiledb_array` object can set a preference for conversion for each retrieved object. This preference can also be encoded in a configuration file as R (version 4.0.0 or later) allows a user- and package specific configuration files. These helper functions set and retrieve the value, respectively, or retrieve the cached value from the package environment where it is set at package load.

Note that the value must be one of 'asis' (the default), 'array', 'matrix' 'data.frame', 'data.table' or 'tibble'. The latter two require the corresponding package to be installed.

Value

For the setter, `TRUE` is returned invisibly but the function is invoked for the side effect of storing the value. For either getter, the character value.

Note

This function requires R version 4.0.0 or later to utilise the per-user config directory accessor function. For older R versions, please set the attribute directly when creating the `tiledb_array` object, or via the `return_as()` method.

---

schema,character-method

Return a schema from a URI character value

Description

Return a schema from a URI character value

Usage

```
## S4 method for signature 'character'
schema(object, ...)
```

Arguments

object   A character variable with a URI
...   Extra parameters such as 'enckey', the encryption key

Value

The scheme for the object

---

schema,tiledb_array-method

Return a schema from a tiledb_array object

Description

Return a schema from a tiledb_array object

Usage

```
## S4 method for signature 'tiledb_array'
schema(object, ...)
```

Arguments

object	tiledb array object
...	Extra parameter for function signature, currently unused

Value

The scheme for the object

---

schema_check   *Check the schema for correctness*

Description

Returns the tiledb_array schema for correctness

Usage

```
schema_check(object)

## S4 method for signature 'tiledb_array_schema'
schema_check(object)

check(object)

## S4 method for signature 'tiledb_array_schema'
check(object)

tiledb_array_schema_check(object)
```

Arguments

object	A TileDB Schema object

Value

The boolean value TRUE is returned for a correct schema; for an incorrect schema an error condition
is triggered.

---

selected_points   *Retrieve selected_points values for the array*

Description

A tiledb_array object can have a range selection for each dimension attribute. This methods
returns the selection value for 'selected_points' and returns a list (with one element per dimension)
of vectors where each row describes one selected points. Alternatively, the list can be named with
the names providing the match to the corresponding dimension.

Usage

```
selected_points(object)

## S4 method for signature 'tiledb_array'
selected_points(object)
```

Arguments

object   A tiledb_array object

Value

A list which can contain a vector for each dimension

---

selected_points<-   *Set selected_points return values for the array*

Description

A tiledb_array object can have a range selection for each dimension attribute. This methods
sets the selection value for 'selected_points' which is a list (with one element per dimension) of
two-column matrices where each row describes one pair of minimum and maximum values. Alter-
natively, the list can be named with the names providing the match to the corresponding dimension.

Usage

```
selected_points(x) <- value

## S4 replacement method for signature 'tiledb_array'
selected_points(x) <- value
```

Arguments

x	A tiledb_array object
value	A list of vectors where each list element 'i' corresponds to the dimension attribute 'i'.

Value

The modified tiledb_array array object

---

selected_ranges	*Retrieve selected_ranges values for the array*

Description

A tiledb_array object can have a range selection for each dimension attribute. This methods returns the selection value for 'selected_ranges' and returns a list (with one element per dimension) of two-column matrices where each row describes one pair of minimum and maximum values. Alternatively, the list can be named with the names providing the match to the corresponding dimension.

Usage

```
selected_ranges(object)

## S4 method for signature 'tiledb_array'
selected_ranges(object)
```

Arguments

object	A tiledb_array object

Value

A list which can contain a matrix for each dimension

---

selected_ranges<-	*Set selected_ranges return values for the array*

Description

A tiledb_array object can have a range selection for each dimension attribute. This methods sets the selection value for 'selected_ranges' which is a list (with one element per dimension) of two-column matrices where each row describes one pair of minimum and maximum values. Alternatively, the list can be named with the names providing the match to the corresponding dimension.

Usage

```
selected_ranges(x) <- value

## S4 replacement method for signature 'tiledb_array'
selected_ranges(x) <- value
```

Arguments

x   A tiledb_array object

value   A list of two-column matrices where each list element 'i' corresponds to the
            dimension attribute 'i'. The matrices can contain rows where each row contains
            the minimum and maximum value of a range.

Value

The modified tiledb_array array object

---

set_max_chunk_size   *Set the filter_list's max_chunk_size*

Description

Set the filter_list's max_chunk_size

Usage

```
set_max_chunk_size(object, value)

## S4 method for signature 'tiledb_filter_list,numeric'
set_max_chunk_size(object, value)

tiledb_filter_list_set_max_chunk_size(object, value)
```

Arguments

object   tiledb_filter_list

value   A numeric value

Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
set_max_chunk_size(filter_list, 10)
```

show,tiledb_array-method

Prints a tiledb_array object

Description

Prints a tiledb_array object

Usage

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

Arguments

object   A tiledb array object

show,tiledb_array_schema-method

Prints an array schema object

Description

Prints an array schema object

Usage

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

Arguments

object   A TileDB Schema object

show,tiledb_attr-method

Prints an attribute object

Description

Prints an attribute object

Usage

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

Arguments

object   An attribute object

show,tiledb_config-method

Prints the config object to STDOUT

Description

Prints the config object to STDOUT

Usage

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

Arguments

object   tiledb_config object

Examples

```
cfg <- tiledb_config()
show(cfg)
```

show,tiledb_dim-method
                              Prints a dimension object

Description

Prints a dimension object

Usage

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

Arguments

object   A `tiledb_dim` object

---

show,tiledb_domain-method
                              Prints a domain object

Description

Prints a domain object

Usage

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

Arguments

object   A domain object

---

show,tiledb_filter-method

Prints a filter object

Description

Prints a filter object

Usage

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

Arguments

object   A filter object

---

show,tiledb_filter_list-method

Prints a filter_list object

Description

Prints a filter_list object

Usage

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

Arguments

object   A filter_list object

show,tiledb_group-method
Display the TileDB Group object to STDOUT

Description

Display the TileDB Group object to STDOUT

Usage

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

Arguments

object   tiledb_group object

---

statusBatched   *Return 'batched' status*

Description

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

Usage

```
statusBatched(obj)
```

Arguments

obj   A list object as returned by createBatched

Value

The Query status as a character variable

strings_as_factors   *Retrieve strings_as_factors conversion toggle*

Description

A `tiledb_array` object containing character column can have those converted to factors variables. This methods returns the selection value for 'strings_as_factors'.

Usage

```
strings_as_factors(object)

## S4 method for signature 'tiledb_array'
strings_as_factors(object)
```

Arguments

object   A `tiledb_array` object

Value

A logical value indicating whether an `strings_as_factors` return is selected

strings_as_factors<-   *Set strings_as_factors return toggle*

Description

A `tiledb_array` object containing character column can have those converted to factors variables. This methods sets the selection value for 'strings_as_factors'.

Usage

```
strings_as_factors(x) <- value

## S4 replacement method for signature 'tiledb_array'
strings_as_factors(x) <- value
```

Arguments

x   A `tiledb_array` object
value   A logical value with the selection

Value

The modified `tiledb_array` array object

```
tdb_collect,tiledb_array-method
```
Collect the query results to finalize piped expression

Description

Collect the query results to finalize piped expression

Usage

```
## S4 method for signature 'tiledb_array'
tdb_collect(x, ...)
```

Arguments

x   A tiledb_array object as first argument, permitting piping

...   Ignored

Value

The object returning from a tiledb_array query (the type of which can be set via the return preference mechanism, see the help for ″[″ accessor)

```
tdb_filter,tiledb_array-method
```
Filter from array for query via logical conditions

Description

Filter from array for query via logical conditions

Usage

```
## S4 method for signature 'tiledb_array'
tdb_filter(x, ..., strict = TRUE)
```

Arguments

x   A tiledb_array object as first argument, permitting piping

...   One or more expressions that are parsed as query_condition objects

strict   A boolean toogle to, if set, errors if a non-existing attribute is selected or filtered on, defaults to 'TRUE'; if 'FALSE' a warning is shown by execution proceeds.

Value

The tiledb_array object, permitting piping

```
tdb_select,tiledb_array-method
```
Select attributes from array for query

Description

Select attributes from array for query

Usage

```
## S4 method for signature 'tiledb_array'
tdb_select(x, ...)
```

Arguments

x	A tiledb_array object as first argument, permitting piping
...	One or more attributes of the query

Value

The tiledb_array object, permitting piping

```
tile,tiledb_dim-method
```
Return the tiledb_dim *tile extent*

Description

Return the tiledb_dim tile extent

Usage

```
## S4 method for signature 'tiledb_dim'
tile(object)
```

Arguments

object	A tiledb_dim object

Value

A scalar tile extent

Examples

```
d1 <- tiledb_dim("d1", domain = c(5L, 10L), tile = 2L)
tile(d1)
```

---

tiledb_array | *Constructs a tiledb_array object backed by a persisted tiledb array uri*

Description

tiledb_array returns a new object. This class is experimental.

Usage

```
tiledb_array(
  uri,
  query_type = c("READ", "WRITE"),
  is.sparse = NA,
  attrs = character(),
  extended = TRUE,
  selected_ranges = list(),
  selected_points = list(),
  query_layout = character(),
  datetimes_as_int64 = FALSE,
  encryption_key = character(),
  query_condition = new("tiledb_query_condition"),
  timestamp_start = as.POSIXct(double(), origin = "1970-01-01"),
  timestamp_end = as.POSIXct(double(), origin = "1970-01-01"),
  return_as = get_return_as_preference(),
  query_statistics = FALSE,
  strings_as_factors = getOption("stringsAsFactors", FALSE),
  keep_open = FALSE,
  sil = list(),
  dumpbuffers = character(),
  buffers = list(),
  ctx = tiledb_get_context(),
  as.data.frame = FALSE
)

tiledb_dense(...)

tiledb_sparse(...)
```

Arguments

uri	uri path to the tiledb dense array
query_type	optionally loads the array in "READ" or "WRITE" only modes.
is.sparse	optional logical switch, defaults to "NA" letting array determine it
attrs	optional character vector to select attributes, default is empty implying all are selected, the special value NA_character_ has the opposite effect and implies no attributes are returned.

extended optional logical switch selecting wide 'data.frame' format, defaults to TRUE

selected_ranges
optional A list with matrices where each matrix i describes the (min,max) pair of ranges selected for dimension i

selected_points
optional A list with vectors where each vector i describes the points selected in dimension i

query_layout optional A value for the TileDB query layout, defaults to an empty character variable indicating no special layout is set

datetimes_as_int64
optional A logical value selecting date and datetime value representation as 'raw' integer64 and not as Date, POSIXct or nanotime objects.

encryption_key optional A character value with an AES-256 encryption key in case the array was written with encryption.

query_condition
optional tiledb_query_condition object, by default uninitialized without a condition; this functionality requires TileDB 2.3.0 or later

timestamp_start
optional A POSIXct Datetime value determining the inclusive time point at which the array is to be openened. No fragments written earlier will be considered.

timestamp_end optional A POSIXct Datetime value determining the inclusive time point until which the array is to be openened. No fragments written earlier later be considered.

return_as optional A character value with the desired tiledb_array conversion, permitted values are 'asis' (default, returning a list of columns), 'array', 'matrix', 'data.frame', 'data.table', 'tibble', 'arrow_table', or 'arrow' (as an alias for 'arrow_table'; here 'data.table', 'tibble' and 'arrow' require the respective packages to be installed. The existing as.* arguments take precedent over this.

query_statistics
optional A logical value, defaults to 'FALSE'; if 'TRUE' the query statistics are returned (as a JSON string) via the attribute 'query_statistics' of the return object.

strings_as_factors
An optional logical to convert character columns to factor type; defaults to the value of getOption("stringsAsFactors", FALSE).

keep_open An optional logical to not close after read or write

sil optional A list, by default empty to store schema information when query objects are parsed.

dumpbuffers An optional character variable with a directory name (relative to /dev/shm) for writing out results buffers (for internal use / testing)

buffers An optional list with full pathnames of shared memory buffers to read data from

ctx optional tiledb_ctx

as.data.frame	An optional deprecated alternative to `return_as="data.frame"` which has been deprecated and removed, but is still used in one BioConductor package; this argument will be removed once the updated package has been released.
...	Used as a pass-through for `tiledb_dense` and `tiledb_sparse` aliasing

Value

tiledb_array object

---

tiledb_array-class *An S4 class for a TileDB Array*

Description

This class replaces the earlier (and now removed) `tiledb_dense` and `tiledb_sparse` and provides equivalent functionality based on a refactored implementation utilising newer TileDB features.

Slots

ctx A TileDB context object

uri A character desprition with the array URI

is.sparse A logical value whether the array is sparse or not

attrs A character vector to select particular column 'attributes'; default is an empty character vector implying 'all' columns, the special value `NA_character_` has the opposite effect and selects 'none'.

extended A logical value, defaults to `TRUE`, indicating whether index columns are returned as well.

selected_ranges An optional list with matrices where each matrix i describes the (min,max) pair of ranges for dimension i

selected_points An optional list with vectors where each vector i describes the selected points for dimension i

query_layout An optional character value

datetimes_as_int64 A logical value

encryption_key A character value

query_condition A Query Condition object

timestamp_start A POSIXct datetime variable for the inclusive interval start

timestamp_end A POSIXct datetime variable for the inclusive interval start

return_as A character value with the desired `tiledb_array` conversion, permitted values are 'asis' (default, returning a list of columns), 'array', 'matrix','data.frame', 'data.table' 'tibble', 'arrow_table' or 'arrow' (where the last two are synomyms); note that 'data.table', 'tibble' and 'arrow' require the respective packages to be installed.

query_statistics A logical value, defaults to 'FALSE'; if 'TRUE' the query statistics are returned (as a JSON string) via the attribute 'query_statistics' of the return object.

sil An optional and internal list object with schema information, used for parsing queries.

dumpbuffers An optional character variable with a directory name (relative to /dev/shm) for writing out results buffers (for internal use / testing)

buffers An optional list with full pathnames of shared memory buffers to read data from

strings_as_factors An optional logical to convert character columns to factor type

keep_open An optional logical to not close after read or write

ptr External pointer to the underlying implementation

---

tiledb_array_apply_aggregate

Run an aggregate query on the given (sparse) array and attribute

---

Description

For dense arrays, use tiledb_query_apply_aggregate after setting an appropriate subarray.

Usage

```
tiledb_array_apply_aggregate(
  array,
  attrname,
  operation = c("Count", "NullCount", "Min", "Max", "Mean", "Sum"),
  nullable = TRUE
)
```

Arguments

array	A TileDB Array object
attrname	The name of an attribute
operation	The name of aggregation operation
nullable	A boolean toggle whether the attribute is nullable

Value

The value of the aggregation

`tiledb_array_close` *Close a TileDB Array*

Description

Close a TileDB Array

Usage

```
tiledb_array_close(arr)
```

Arguments

arr   A TileDB Array object as for example returned by `tiledb_array()`

Value

The TileDB Array object but closed

`tiledb_array_create` *Creates a new TileDB array given an input schema.*

Description

Creates a new TileDB array given an input schema.

Usage

```
tiledb_array_create(uri, schema, encryption_key)
```

Arguments

uri   URI specifying path to create the TileDB array object
schema   tiledb_array_schema object
encryption_key optional A character value with an AES-256 encryption key in case the array
                should be encrypted.

Examples

```
## Not run:
pth <- tempdir()
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32")))
tiledb_array_create(pth, sch)
tiledb_object_type(pth)

## End(Not run)
```

tiledb_array_delete_fragments
                          Delete fragments written between the start and end times given

Description

Delete fragments written between the start and end times given

Usage

```
tiledb_array_delete_fragments(
  arr,
  ts_start,
  ts_end,
  ctx = tiledb_get_context()
)
```

Arguments

arr	A TileDB Array object as for example returned by tiledb_array()
ts_start	A Datetime object that will be converted to millisecond granularity
ts_end	A Datetime object that will be converted to millisecond granularity
ctx	A tiledb_ctx object (optional)

Value

A boolean indicating success

tiledb_array_delete_fragments_list
                          Delete fragments written given by their URIs

Description

Delete fragments written given by their URIs

Usage

```
tiledb_array_delete_fragments_list(arr, fragments, ctx = tiledb_get_context())
```

Arguments

arr	A TileDB Array object as for example returned by tiledb_array()
fragments	A character vector with fragment URIs
ctx	A tiledb_ctx object (optional)

Value

A boolean indicating success

---

tiledb_array_get_non_empty_domain_from_index
Get the non-empty domain from a TileDB Array by index

Description

This functions works for both fixed- and variable-sized dimensions and switches internally.

Usage

```
tiledb_array_get_non_empty_domain_from_index(arr, idx)
```

Arguments

arr   A TileDB Array

idx   An integer index between one the number of dimensions

Value

A two-element object is returned describing the domain of selected dimension; it will either be a numeric vector in case of a fixed-size fixed-sized dimensions, or a character vector for a variable-sized one.

---

tiledb_array_get_non_empty_domain_from_name
Get the non-empty domain from a TileDB Array by name

Description

This functions works for both fixed- and variable-sized dimensions and switches internally.

Usage

```
tiledb_array_get_non_empty_domain_from_name(arr, name)
```

Arguments

arr   A TileDB Array

name   An character variable with a dimension name

Value

A two-element object is returned describing the domain of selected dimension; it will either be a numeric vector in case of a fixed-size fixed-sized dimensions, or a character vector for a variable-sized one.

`tiledb_array_has_enumeration`

Check for Enumeration (aka Factor aka Dictionary)

Description

Check for Enumeration (aka Factor aka Dictionary)

Usage

```
tiledb_array_has_enumeration(arr)
```

Arguments

arr   A TileDB Array object

Value

A boolean indicating if the array has homogeneous domains

`tiledb_array_is_heterogeneous`

Check for Heterogeneous Domain

Description

Check for Heterogeneous Domain

Usage

```
tiledb_array_is_heterogeneous(arr)
```

Arguments

arr   A TileDB Array object

Value

A boolean indicating if the array has heterogenous domains

---

tiledb_array_is_homogeneous

Check for Homogeneous Domain

Description

Check for Homogeneous Domain

Usage

```
tiledb_array_is_homogeneous(arr)
```

Arguments

arr   A TileDB Array object

Value

A boolean indicating if the array has homogeneous domains

---

tiledb_array_is_open   *Test if TileDB Array is open*

Description

Test if TileDB Array is open

Usage

```
tiledb_array_is_open(arr)
```

Arguments

arr   A TileDB Array object as for example returned by tiledb_array()

Value

A boolean indicating whether the TileDB Array object is open

`tiledb_array_is_open_for_reading`

Test if TileDB Array is open for reading

Description

Test if TileDB Array is open for reading

Usage

```
tiledb_array_is_open_for_reading(arr)
```

Arguments

arr   A TileDB Array object as for example returned by `tiledb_array()`

Value

A boolean indicating whether the TileDB Array object is open, and the mode is "READ".

`tiledb_array_is_open_for_writing`

Test if TileDB Array is open for writing

Description

Test if TileDB Array is open for writing

Usage

```
tiledb_array_is_open_for_writing(arr)
```

Arguments

arr   A TileDB Array object as for example returned by `tiledb_array()`

Value

A boolean indicating whether the TileDB Array object is open, and the mode is "WRITE".

---

tiledb_array_open *Open a TileDB Array*

Description

Open a TileDB Array

Usage

```
tiledb_array_open(
  arr,
  type = if (tiledb_version(TRUE) >= "2.12.0") {
     c("READ", "WRITE", "DELETE",
    "MODIFY_EXCLUSIVE")
 } else {
     c("READ", "WRITE")
 }
)
```

Arguments

arr	A TileDB Array object as for example returned by `tiledb_array()`
type	A character value that must be either 'READ', 'WRITE' or (for TileDB 2.12.0 or later) 'DELETE' or 'MODIFY_EXCLUSIVE'

Value

The TileDB Array object but opened for reading or writing

---

tiledb_array_open_at *Open a TileDB Array at Timestamp*

Description

Open a TileDB Array at Timestamp

Usage

```
tiledb_array_open_at(arr, type = c("READ", "WRITE"), timestamp)
```

Arguments

arr	A TileDB Array object as for example returned by `tiledb_array()`
type	A character value that must be either 'READ' or 'WRITE'
timestamp	A Datetime object that will be converted to millisecond granularity

Value

The TileDB Array object but opened for reading or writing

---

tiledb_array_schema   *Constructs a* tiledb_array_schema *object*

Description

Constructs a tiledb_array_schema object

Usage

```
tiledb_array_schema(
  domain,
  attrs,
  cell_order = "COL_MAJOR",
  tile_order = "COL_MAJOR",
  sparse = FALSE,
  coords_filter_list = NULL,
  offsets_filter_list = NULL,
  validity_filter_list = NULL,
  capacity = 10000L,
  allows_dups = FALSE,
  enumerations = NULL,
  ctx = tiledb_get_context()
)
```

Arguments

domain	tiledb_domain object
attrs	a list of one or more tiledb_attr objects
cell_order	(default "COL_MAJOR")
tile_order	(default "COL_MAJOR")
sparse	(default FALSE)
coords_filter_list	(optional)
offsets_filter_list	(optional)
validity_filter_list	(optional)
capacity	(optional)
allows_dups	(optional, requires 'sparse' to be TRUE)
enumerations	(optional) named list of enumerations
ctx	tiledb_ctx object (optional)

Value

An object of class `tiledb_array_schema`.

Examples

```
schema <- tiledb_array_schema(
  dom = tiledb_domain(
    dims = c(
      tiledb_dim("rows", c(1L, 4L), 4L, "INT32"),
      tiledb_dim("cols", c(1L, 4L), 4L, "INT32")
    )
  ),
  attrs = c(tiledb_attr("a", type = "INT32")),
  cell_order = "COL_MAJOR",
  tile_order = "COL_MAJOR",
  sparse = FALSE
)
schema
```

`tiledb_array_schema-class`

An S4 class for the TileDB array schema

Description

An S4 class for the TileDB array schema

Slots

`ptr` An external pointer to the underlying implementation

`arrptr` An optional external pointer to the underlying array, or NULL if missing

---

`tiledb_array_schema_evolution`

Creates a 'tiledb_array_schema_evolution' object

Description

Creates a 'tiledb_array_schema_evolution' object

Usage

```
tiledb_array_schema_evolution(ctx = tiledb_get_context())
```

Arguments

ctx   (optional) A TileDB Ctx object; if not supplied the default context object is
                retrieved

Value

A 'array_schema_evolution' object

---

tiledb_array_schema_evolution-class
                        An S4 class for a TileDB ArraySchemaEvolution object

---

Description

An S4 class for a TileDB ArraySchemaEvolution object

Slots

ptr An external pointer to the underlying implementation

---

tiledb_array_schema_evolution_add_attribute
                        Add an Attribute to a TileDB Array Schema Evolution object

---

Description

Add an Attribute to a TileDB Array Schema Evolution object

Usage

```
tiledb_array_schema_evolution_add_attribute(object, attr)
```

Arguments

object   A TileDB 'array_schema_evolution' object
attr   A TileDB attribute

Value

The modified 'array_schema_evolution' object, invisibly

tiledb_array_schema_evolution_add_enumeration

Add an Enumeration to a TileDB Array Schema Evolution object

Description

Add an Enumeration to a TileDB Array Schema Evolution object

Usage

```
tiledb_array_schema_evolution_add_enumeration(
  object,
  name,
  enums,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

Arguments

object	A TileDB 'array_schema_evolution' object
name	A character value with the name for the Enumeration
enums	A character vector
ordered	(optional) A boolean switch whether the enumeration is ordered
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

The modified 'array_schema_evolution' object, invisibly

tiledb_array_schema_evolution_add_enumeration_empty

Evolve an Array Schema by adding an empty Enumeration

Description

Evolve an Array Schema by adding an empty Enumeration

Usage

```
tiledb_array_schema_evolution_add_enumeration_empty(
  ase,
  enum_name,
  type_str = "ASCII",
  cell_val_num = NA_integer_,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

Arguments

ase	An ArraySchemaEvolution object
enum_name	A character value with the Enumeration name
type_str	A character value with the TileDB type, defaults to 'ASCII'
cell_val_num	An integer with number values per cell, defaults to NA_integer_ to flag the NA value use for character values
ordered	A logical value indicating standard factor (when FALSE, the default) or ordered (when TRUE)
ctx	Optional tiledb_ctx object

---

tiledb_array_schema_evolution_array_evolve
Evolve an Array Schema

Description

Evolve an Array Schema

Usage

```
tiledb_array_schema_evolution_array_evolve(object, uri)
```

Arguments

object	A TileDB 'array_schema_evolution' object
uri	A character variable with an URI

Value

The modified 'array_schema_evolution' object, invisibly

tiledb_array_schema_evolution_drop_attribute

Drop an attribute given by name from a TileDB Array Schema Evolution object

Description

Drop an attribute given by name from a TileDB Array Schema Evolution object

Usage

```
tiledb_array_schema_evolution_drop_attribute(object, attrname)
```

Arguments

object	A TileDB 'array_schema_evolution' object
attrname	A character variable with an attribute name

Value

The modified 'array_schema_evolution' object, invisibly

tiledb_array_schema_evolution_drop_enumeration

Drop an Enumeration given by name from a TileDB Array Schema Evolution object

Description

Drop an Enumeration given by name from a TileDB Array Schema Evolution object

Usage

```
tiledb_array_schema_evolution_drop_enumeration(object, attrname)
```

Arguments

object	A TileDB 'array_schema_evolution' object
attrname	A character variable with an attribute name

Value

The modified 'array_schema_evolution' object, invisibly

---

tiledb_array_schema_evolution_expand_current_domain
                    Expand an the Current Domain of an Array via Array Schema Evolution

---

Description

Expand an the Current Domain of an Array via Array Schema Evolution

Usage

```
tiledb_array_schema_evolution_expand_current_domain(ase, cd)
```

Arguments

ase   An ArraySchemaEvolution object

cd   A CurrentDomain object

Value

The modified ArraySchemaEvolution object

---

tiledb_array_schema_evolution_extend_enumeration
                    Extend an Evolution via Array Schema Evolution

---

Description

Extend an Evolution via Array Schema Evolution

Usage

```
tiledb_array_schema_evolution_extend_enumeration(
  ase,
  array,
  enum_name,
  new_values,
  nullable = FALSE,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

Arguments

ase	An ArraySchemaEvolution object
array	A TileDB Array object
enum_name	A character value with the Enumeration name
new_values	A character vector with the new Enumeration values
nullable	A logical value indicating if the Enumeration can contain missing values (with a default of FALSE)
ordered	A logical value indicating standard factor (when FALSE, the default) or ordered (when TRUE)
ctx	Optional tiledb_ctx object

Value

The modified ArraySchemaEvolution object

---

tiledb_array_schema_get_current_domain

Get the Current Domain of an Array Schema

Description

Note that tiledb_current_domain object may be empty.

Usage

```
tiledb_array_schema_get_current_domain(schema, ctx = tiledb_get_context())
```

Arguments

schema	A TileDB Schema object
ctx	Optional tiledb_ctx object

Value

An object of class tiledb_current_domain.

```
tiledb_array_schema_set_coords_filter_list
```
Set a Filter List for Coordinate of a TileDB Schema

Description

Set a Filter List for Coordinate of a TileDB Schema

Usage

```
tiledb_array_schema_set_coords_filter_list(sch, fl)
```

Arguments

sch	A TileDB Schema object
fl	A TileDB Filter List object

Value

The modified `tiledb_array_schema` object.

```
tiledb_array_schema_set_current_domain
```
Set a Current Domain of an Array Schema

Description

Set a Current Domain of an Array Schema

Usage

```
tiledb_array_schema_set_current_domain(schema, cd, ctx = tiledb_get_context())
```

Arguments

schema	A TileDB Schema object
cd	A TileDB Current Domain object
ctx	Optional tiledb_ctx object

Value

Nothing is returned from this function (but an error, should it occur is reported)

tiledb_array_schema_set_enumeration_empty

Add an empty Enumeration to a Schema

Description

Add an empty Enumeration to a Schema

Usage

```
tiledb_array_schema_set_enumeration_empty(
  schema,
  attr,
  enum_name,
  type_str = "ASCII",
  cell_val_num = NA_integer_,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

Arguments

schema	A TileDB Schema object
attr	An Attribute for which an empty Enumeration will be added
enum_name	A character value with the Enumeration name
type_str	A character value with the TileDB type, defaults to 'ASCII'
cell_val_num	An integer with number values per cell, defaults to NA_integer_ to flag the NA value use for character values
ordered	A logical value indicated standard factor (when FALSE, the default) or ordered (when TRUE)
ctx	Optional tiledb_ctx object

Value

The modified tiledb_array_schema object.

`tiledb_array_schema_set_offsets_filter_list`
                              Set a Filter List for Variable-Sized Offsets of a TileDB Schema

---

Description

Set a Filter List for Variable-Sized Offsets of a TileDB Schema

Usage

```
tiledb_array_schema_set_offsets_filter_list(sch, fl)
```

Arguments

sch   A TileDB Schema object

fl   A TileDB Filter List object

Value

The modified `tiledb_array_schema` object.

`tiledb_array_schema_set_validity_filter_list`
                              Set a Filter List for Validity of a TileDB Schema

---

Description

Set a Filter List for Validity of a TileDB Schema

Usage

```
tiledb_array_schema_set_validity_filter_list(sch, fl)
```

Arguments

sch   A TileDB Schema object

fl   A TileDB Filter List object

Value

The modified `tiledb_array_schema` object.

---

tiledb_array_schema_version

Check the version of the array schema

Description

Returns the (internal) version of the `tiledb_array` schema

Usage

```
tiledb_array_schema_version(object)
```

Arguments

object   A TileDB Schema object

Value

An integer value describing the internal schema format version

---

tiledb_array_upgrade_version

Upgrade an Array to the current TileDB Array Schema Format

Description

Upgrade an Array to the current TileDB Array Schema Format

Usage

```
tiledb_array_upgrade_version(array, config = NULL, ctx = tiledb_get_context())
```

Arguments

array   A TileDB Array object

config   A TileDB Configuration (optional, default NULL)

ctx   A tiledb_ctx object (optional)

Value

Nothing is returned as the function is invoked for its side effect

---

`tiledb_arrow_array_ptr`

(Deprecated) Allocate (or Release) Arrow Array and Schema Pointers

---

Description

These functions allocate (and free) appropriate pointer objects for, respectively, Arrow array and schema objects. These functions are deprecated and will be removed, it is recommended to rely directly on the `nanoarrow` replacements.

Usage

```
tiledb_arrow_array_ptr()

tiledb_arrow_schema_ptr()

tiledb_arrow_array_del(ptr)

tiledb_arrow_schema_del(ptr)
```

Arguments

ptr   A external pointer object previously allocated with these functions

Value

The allocating functions return the requested pointer

`tiledb_attr`   *Constructs a* `tiledb_attr` *object*

Description

Constructs a `tiledb_attr` object

Usage

```
tiledb_attr(
  name,
  type,
  filter_list = tiledb_filter_list(),
  ncells = 1,
  nullable = FALSE,
  enumeration = NULL,
  ctx = tiledb_get_context()
)
```

Arguments

name	The dimension name / label string; if missing default `""` is used.
type	The tiledb_attr TileDB datatype string; if missing the user is alerted that this is a *required* parameter.
filter_list	(default filter_list("NONE")) An optional tiledb_filter_list object
ncells	(default 1) The number of cells, use `NA` to signal variable length
nullable	(default FALSE) A logical switch whether the attribute can have missing values
enumeration	(default NULL) A character vector of dictionary values
ctx	tiledb_ctx object (optional)

Value

`tiledb_dim` object

Examples

```
flt <- tiledb_filter_list(list(tiledb_filter("GZIP")))
attr <- tiledb_attr(
  name = "a1", type = "INT32",
  filter_list = flt
)
attr
```

---

tiledb_attr-class   *An S4 class for a TileDB attribute*

Description

An S4 class for a TileDB attribute

Slots

`ptr` External pointer to the underlying implementation

`tiledb_attribute_get_cell_size`

Get the TileDB Attribute cell size

Description

Get the TileDB Attribute cell size

Usage

```
tiledb_attribute_get_cell_size(attr)
```

Arguments

attr   A TileDB Attribute object

Value

A numeric value with the cell size

`tiledb_attribute_get_enumeration`

Get the TileDB Attribute Enumeration

Description

Get the TileDB Attribute Enumeration

Usage

```
tiledb_attribute_get_enumeration(attr, arr, ctx = tiledb_get_context())

tiledb_attribute_get_enumeration_ptr(attr, arrptr, ctx = tiledb_get_context())
```

Arguments

attr	A TileDB Attribute object
arr	A Tiledb Array object
ctx	A Tiledb Context object (optional)
arrptr	A Tiledb Array object pointer

Value

A character vector with the enumeration (of length zero if none)

tiledb_attribute_get_fill_value
Get the fill value for a TileDB Attribute

Description

Get the fill value for a TileDB Attribute

Usage

```
tiledb_attribute_get_fill_value(attr)
```

Arguments

attr   A TileDB Attribute object

Value

The fill value for the attribute

tiledb_attribute_get_nullable
Get the TileDB Attribute Nullable flag value

Description

Get the TileDB Attribute Nullable flag value

Usage

```
tiledb_attribute_get_nullable(attr)
```

Arguments

attr   A TileDB Attribute object

Value

A boolean value with the 'Nullable' status

---

tiledb_attribute_has_enumeration

Test if TileDB Attribute has an Enumeration

Description

Test if TileDB Attribute has an Enumeration

Usage

```
tiledb_attribute_has_enumeration(attr, ctx = tiledb_get_context())
```

Arguments

attr	A TileDB Attribute object
ctx	A Tiledb Context object (optional)

Value

A logical value indicating if the attribute has an enumeration

---

tiledb_attribute_is_ordered_enumeration_ptr

Check if TileDB Attribute Enumeration is Ordered

Description

Check if TileDB Attribute Enumeration is Ordered

Usage

```
tiledb_attribute_is_ordered_enumeration_ptr(
  attr,
  arrptr,
  ctx = tiledb_get_context()
)
```

Arguments

attr	A Tiledb Array object
arrptr	A Tiledb Array object pointer
ctx	A Tiledb Context object (optional)

Value

A character vector with the enumeration (of length zero if none)

```
tiledb_attribute_is_variable_sized
```
Check whether TileDB Attribute is variable-sized

Description

Check whether TileDB Attribute is variable-sized

Usage

```
tiledb_attribute_is_variable_sized(attr)
```

Arguments

attr   A TileDB Attribute object

Value

A boolean value indicating variable-size or not

```
tiledb_attribute_set_enumeration_name
```
Set a TileDB Attribute Enumeration Name

Description

Set a TileDB Attribute Enumeration Name

Usage

```
tiledb_attribute_set_enumeration_name(
  attr,
  enum_name,
  ctx = tiledb_get_context()
)
```

Arguments

attr   A TileDB Attribute object

enum_name A character value with the enumeration value

ctx   A Tiledb Context object (optional)

Value

The modified TileDB Attribute object

---

tiledb_attribute_set_fill_value

Set the fill value for a TileDB Attribute

Description

Set the fill value for a TileDB Attribute

Usage

```
tiledb_attribute_set_fill_value(attr, value)
```

Arguments

attr	A TileDB Attribute object
value	A fill value

Value

NULL is returned invisibly

---

tiledb_attribute_set_nullable

Set the TileDB Attribute Nullable flags

Description

Set the TileDB Attribute Nullable flags

Usage

```
tiledb_attribute_set_nullable(attr, flag)
```

Arguments

attr	A TileDB Attribute object
flag	A boolean flag to turn 'Nullable' on or off

Value

Nothing is returned

---

tiledb_config *Creates a* tiledb_config *object*

Description

Note that for actually setting persistent values, the (altered) config object needs to used to create (or update) the tiledb_ctx object. Similarly, to check whether values are set, one should use the config method of the of the tiledb_ctx object. Examples for this are ctx <- tiledb_ctx(limitTileDBCores()) to use updated configuration values to create a context object, and cfg <- config(ctx) to retrieve it.

Usage

```
tiledb_config(config = NA_character_)
```

Arguments

config   (optional) character vector of config parameter names, values

Value

tiledb_config object

Examples

```
cfg <- tiledb_config()
cfg["sm.tile_cache_size"]

# set tile cache size to custom value
cfg <- tiledb_config(c("sm.tile_cache_size" = "100"))
cfg["sm.tile_cache_size"]
```

---

tiledb_config-class *An S4 class for a TileDB configuration*

Description

An S4 class for a TileDB configuration

Slots

ptr An external pointer to the underlying implementation

---

tiledb_config_as_built_json

Return the 'AsBuilt' JSON string

Description

Return the 'AsBuilt' JSON string

Usage

```
tiledb_config_as_built_json()
```

Value

The JSON string containing 'AsBuilt' information

Examples

```
if (tiledb_version(TRUE) > "2.17") {
  txt <- tiledb::tiledb_config_as_built_json()
}
## now eg either one of
##   sapply(jsonlite::fromJSON(txt)$as_built$parameters$storage_backends, \(x) x[[1]])
##   sapply(RcppSimdJson::fparse(txt)$as_built$parameters$storage_backends, \(x) x[[1]])
## will return a named vector such as
##   c(azure = FALSE, gcs = FALSE, hdfs = FALSE, s3 = TRUE)
```

---

tiledb_config_as_built_show

Display the 'AsBuilt' JSON string

Description

Display the 'AsBuilt' JSON string

Usage

```
tiledb_config_as_built_show()
```

Value

Nothing is returned but as a side-effect the 'AsBuilt' string is displayed

tiledb_config_load   *Load a saved* tiledb_config *file from disk*

Description

Load a saved tiledb_config file from disk

Usage

```
tiledb_config_load(path)
```

Arguments

path   path to the config file

Examples

```
tmp <- tempfile()
cfg <- tiledb_config(c("sm.tile_cache_size" = "10"))
pth <- tiledb_config_save(cfg, tmp)
cfg <- tiledb_config_load(pth)
cfg["sm.tile_cache_size"]
```

tiledb_config_save   *Save a* tiledb_config *object ot a local text file*

Description

Save a tiledb_config object ot a local text file

Usage

```
tiledb_config_save(config, path)
```

Arguments

config   The tiledb_config object

path   The path to config file to be created

Value

path to created config file

Examples

```
tmp <- tempfile()
cfg <- tiledb_config(c("sm.tile_cache_size" = "10"))
pth <- tiledb_config_save(cfg, tmp)

cat(readLines(pth), sep = "\n")
```

---

tiledb_config_unset   *Unset a TileDB Config parameter to its default value*

Description

Unset a TileDB Config parameter to its default value

Usage

```
tiledb_config_unset(config, param)
```

Arguments

config	A TileDB Config object
param	A character variable with the parameter name

Value

The modified TileDB Config object

---

tiledb_ctx   *Creates a* tiledb_ctx *object*

Description

Creates a tiledb_ctx object

Usage

```
tiledb_ctx(config = NULL, cached = TRUE)
```

Arguments

config	(optional) character vector of config parameter names, values
cached	(optional) logical switch to force new creation

Value

tiledb_ctx object

Examples

```
# default configuration
ctx <- tiledb_ctx()

# optionally set config parameters
ctx <- tiledb_ctx(c("sm.tile_cache_size" = "100"))
```

---

tiledb_ctx-class   *An S4 class for a TileDB context*

Description

An S4 class for a TileDB context

Slots

ptr An external pointer to the underlying implementation

---

tiledb_ctx_set_default_tags
                        Sets default context tags

Description

Sets default context tags

Usage

```
tiledb_ctx_set_default_tags(object)
```

Arguments

object   tiledb_ctx object

---

tiledb_ctx_set_tag   *Sets a string:string "tag" on the Ctx*

Description

Sets a string:string "tag" on the Ctx

Usage

```
tiledb_ctx_set_tag(object, key, value)
```

Arguments

object	tiledb_ctx object
key	string
value	string

Examples

```
ctx <- tiledb_ctx(c("sm.tile_cache_size" = "10"))
cfg <- tiledb_ctx_set_tag(ctx, "tag", "value")
```

---

tiledb_ctx_stats   *Return context statistics as a JSON string*

Description

Return context statistics as a JSON string

Usage

```
tiledb_ctx_stats(object = tiledb_get_context())
```

Arguments

object	A tiledb_ctx object

Value

A JSON-formatted string with context statistics

---

tiledb_current_domain   *Creates a* tiledb_current_domain *object*

Description

Creates a tiledb_current_domain object

Usage

```
tiledb_current_domain(ctx = tiledb_get_context())
```

Arguments

ctx   (optional) A TileDB Ctx object

Value

The tiledb_current_domain object

Examples

```
if (tiledb_version(TRUE) >= "2.25.0") {
  cd <- tiledb_current_domain()
}
```

---

tiledb_current_domain-class
                        An S4 class for a TileDB CurrentDomain object

---

Description

An S4 class for a TileDB CurrentDomain object

Slots

ptr An external pointer to the underlying CurrentDomain object

---

tiledb_current_domain_get_ndrectangle

Get a tiledb_ndrectangle *from a* tiledb_current_domain *object*

Description

Get a tiledb_ndrectangle from a tiledb_current_domain object

Usage

```
tiledb_current_domain_get_ndrectangle(cd)
```

Arguments

cd   A TileDB CurrentDomain object

Value

The corresponding TileDB NDRectangle object

---

tiledb_current_domain_get_type

Get tiledb_current_domain *data type as string*

Description

Get tiledb_current_domain data type as string

Usage

```
tiledb_current_domain_get_type(cd)
```

Arguments

cd   A TileDB CurrentDomain object

Value

The datatype (as string) of the tiledb_current_domain object

tiledb_current_domain_is_empty

Test tiledb_current_domain *object for being empty*

Description

Test tiledb_current_domain object for being empty

Usage

```
tiledb_current_domain_is_empty(cd)
```

Arguments

cd   A TileDB CurrentDomain object

Value

A boolean indicating whether the object is empty or not

tiledb_current_domain_set_ndrectangle

Set a tiledb_ndrectangle *in a* tiledb_current_domain *object*

Description

Set a tiledb_ndrectangle in a tiledb_current_domain object

Usage

```
tiledb_current_domain_set_ndrectangle(cd, ndr)
```

Arguments

cd   A TileDB CurrentDomain object
ndr   A TileDB NDRectangle object

Value

The modifiled TileDB CurrendDomain object

---

tiledb_datatype_R_type

Map from TileDB type to R datatype

Description

This function maps from the TileDB types to the (fewer) key datatypes in R. This can be lossy as TileDB integers range from (signed and unsigned) 8 to 64 bit whereas R only has (signed) 32 bit values. Similarly, R only has 64 bit doubles whereas TileDB has 32 and 64 bit floating point types. TileDB also has more character encodings, and the full range of (NumPy) date and time types.

Usage

```
tiledb_datatype_R_type(datatype)
```

Arguments

datatype   A string describing one TileDB datatype

Value

A string describing the closest match for an R datatype

---

tiledb_delete_metadata

Delete a TileDB Array Metadata object given by key

Description

Delete a TileDB Array Metadata object given by key

Usage

```
tiledb_delete_metadata(arr, key)
```

Arguments

arr   A TileDB Array object

key   A character value describing a metadata key

Value

A boolean indicating success

tiledb_dim	*Constructs a* `tiledb_dim` *object*

Description

Constructs a `tiledb_dim` object

Usage

```
tiledb_dim(
  name,
  domain,
  tile,
  type,
  filter_list = tiledb_filter_list(),
  ctx = tiledb_get_context()
)
```

Arguments

name	The dimension name / label string. This argument is required.
domain	The dimension (inclusive) domain. The domain of a dimension is defined by a (lower bound, upper bound) vector. For type `ASCII`, `NULL` is expected.
tile	The tile dimension tile extent. For type `ASCII`, `NULL` is expected.
type	The dimension TileDB datatype string.
filter_list	An optional `tiledb_filter_list` object, default is no filter
ctx	tiledb_ctx object (optional)

Value

A `tiledb_dim` object

Examples

```
tiledb_dim(name = "d1", domain = c(1L, 10L), tile = 5L, type = "INT32")
```

tiledb_dim-class	*An S4 class for a TileDB dimension object*

Description

An S4 class for a TileDB dimension object

Slots

`ptr` An external pointer to the underlying implementation

---

tiledb_domain   *Constructs a* tiledb_domain *object*

Description

All tiledb_dim must be of the same TileDB type.

Usage

```
tiledb_domain(dims, ctx = tiledb_get_context())
```

Arguments

dims   list() of tiledb_dim objects

ctx   tiledb_ctx (optional)

Value

tiledb_domain

Examples

```
dom <- tiledb_domain(dims = c(
  tiledb_dim("d1", c(1L, 100L), type = "INT32"),
  tiledb_dim("d2", c(1L, 50L), type = "INT32")
))
```

---

tiledb_domain-class   *An S4 class for a TileDB domain*

Description

An S4 class for a TileDB domain

Slots

ptr External pointer to the underlying implementation

tiledb_domain_get_dimension_from_index

Returns a Dimension indicated by index for the given TileDB Domain

Description

Returns a Dimension indicated by index for the given TileDB Domain

Usage

```
tiledb_domain_get_dimension_from_index(domain, idx)
```

Arguments

domain	TileDB Domain object
idx	Integer index of the selected dimension

Value

TileDB Dimension object

tiledb_domain_get_dimension_from_name

Returns a Dimension indicated by name for the given TileDB Domain

Description

Returns a Dimension indicated by name for the given TileDB Domain

Usage

```
tiledb_domain_get_dimension_from_name(domain, name)
```

Arguments

domain	TileDB Domain object
name	A character variable with a dimension name

Value

TileDB Dimension object

tiledb_domain_has_dimension

Check a domain for a given dimension name

Description

Check a domain for a given dimension name

Usage

```
tiledb_domain_has_dimension(domain, name)
```

Arguments

domain	A domain of a TileDB Array schema
name	A character variable with a dimension name

Value

A boolean value indicating if the dimension exists in the domain

tiledb_error_message *Return the error message for a given context*

Description

Note that this function requires an actual error to have occurred.

Usage

```
tiledb_error_message(ctx = tiledb_get_context())
```

Arguments

ctx	A tiledb_ctx object

Value

A character variable with the error message

```
tiledb_filestore_buffer_export
```
Export from a TileDB Filestore to a character variable

Description

Export from a TileDB Filestore to a character variable

Usage

```
tiledb_filestore_buffer_export(
  filestore_uri,
  offset,
  bytes,
  ctx = tiledb_get_context()
)
```

Arguments

filestore_uri	Character with an TileDB Array Schema URI
offset	(optional) Numeric variable with offset from beginnig, default is zero
bytes	(optional) Numeric variable with number of bytes to read, default is zero
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

A character variable containing the filestore content (subject to offset and bytes) is returned

```
tiledb_filestore_buffer_import
```
Import size bytes from a string into a TileDB Filestore

Description

Import size bytes from a string into a TileDB Filestore

Usage

```
tiledb_filestore_buffer_import(
  filestore_uri,
  buf,
  bytes,
  ctx = tiledb_get_context()
)
```

Arguments

`filestore_uri`	Character with an TileDB Array Schema URI
`buf`	Character variable with content to be imported
`bytes`	Number of bytes to be import, defaults to length of buf
`ctx`	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

A boolean is returned to indicate successful completion

`tiledb_filestore_schema_create`

Create an array schema from a given URI with schema

Description

Create an array schema from a given URI with schema

Usage

```
tiledb_filestore_schema_create(uri = NULL, ctx = tiledb_get_context())
```

Arguments

`uri`	Character with an TileDB Array Schema URI, if missing or NULL a default schema is returned
`ctx`	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

An ArraySchema object corresponding to the supplied schema, or a default if missing

---

tiledb_filestore_size   *Return (uncompressed) TileDB Filestore size*

Description

Return (uncompressed) TileDB Filestore size

Usage

```
tiledb_filestore_size(filestore_uri, ctx = tiledb_get_context())
```

Arguments

filestore_uri	Character with an TileDB Array Schema URI
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

A numeric with the size is returned

---

tiledb_filestore_uri_export

Export a file from a TileDB Filestore

Description

Export a file from a TileDB Filestore

Usage

```
tiledb_filestore_uri_export(
  file_uri,
  filestore_uri,
  ctx = tiledb_get_context()
)
```

Arguments

file_uri	Character with a file URI
filestore_uri	Character with an TileDB Array Schema URI
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

A boolean is returned to indicate successful completion

---

tiledb_filestore_uri_import

Import a file into a TileDB Filestore

Description

Import a file into a TileDB Filestore

Usage

```
tiledb_filestore_uri_import(
  filestore_uri,
  file_uri,
  ctx = tiledb_get_context()
)
```

Arguments

filestore_uri	Character with an TileDB Array Schema URI
file_uri	Character with a file URI
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

A boolean is returned to indicate successful completion

---

tiledb_filter   *Constructs a* tiledb_filter *object*

Description

Available filters:

- "NONE"
- "GZIP"
- "ZSTD"
- "LZ4"
- "RLE"
- "BZIP2"
- "DOUBLE_DELTA"
- "BIT_WIDTH_REDUCTION"
- "BITSHUFFLE"

- "BYTESHUFFLE"
- "POSITIVE_DELTA"
- "CHECKSUM_MD5"
- "CHECKSUM_SHA256"
- "DICTIONARY"
- "SCALE_FLOAT" (TileDB 2.11.0 or later)
- "FILTER_XOR" (TileDB 2.12.0 or later)

Usage

```
tiledb_filter(name = "NONE", ctx = tiledb_get_context())
```

Arguments

name	(default "NONE") TileDB filter name string
ctx	tiledb_ctx object (optional)

Details

Valid compression options vary depending on the filter used, consult the TileDB docs for more information.

Value

tiledb_filter object

Examples

```
tiledb_filter("ZSTD")
```

---

tiledb_filter-class *An S4 class for a TileDB filter*

Description

An S4 class for a TileDB filter

Slots

ptr External pointer to the underlying implementation

`tiledb_filter_get_option`

Returns the filter's option

Description

Returns the filter's option

Usage

```
tiledb_filter_get_option(object, option)
```

Arguments

object	tiledb_filter
option	string

Value

Integer value

Examples

```
c <- tiledb_filter("ZSTD")
tiledb_filter_set_option(c, "COMPRESSION_LEVEL", 5)
tiledb_filter_get_option(c, "COMPRESSION_LEVEL")
```

`tiledb_filter_list`   *Constructs a* `tiledb_filter_list` *object*

Description

Constructs a `tiledb_filter_list` object

Usage

```
tiledb_filter_list(filters = c(), ctx = tiledb_get_context())
```

Arguments

filters	an optional list of one or more tiledb_filter_list objects
ctx	tiledb_ctx object (optional)

Value

tiledb_filter_list object

Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
filter_list
```

---

tiledb_filter_list-class

An S4 class for a TileDB filter list

Description

An S4 class for a TileDB filter list

Slots

ptr An external pointer to the underlying implementation

---

tiledb_filter_set_option

Set the option for a filter

Description

Set the option for a filter

Usage

```
tiledb_filter_set_option(object, option, value)
```

Arguments

object	tiledb_filter
option	string
value	int

Value

The modified filter object is returned.

Examples

```
c <- tiledb_filter("ZSTD")
tiledb_filter_set_option(c, "COMPRESSION_LEVEL", 5)
tiledb_filter_get_option(c, "COMPRESSION_LEVEL")
```

---

tiledb_filter_type   *Returns the type of the filter used*

Description

Returns the type of the filter used

Usage

```
tiledb_filter_type(object)
```

Arguments

object   tiledb_filter

Value

TileDB filter type string

Examples

```
c <- tiledb_filter("ZSTD")
tiledb_filter_type(c)
```

---

tiledb_fragment_info   *Constructs a* tiledb_fragment_info *object*

Description

Constructs a tiledb_fragment_info object

Usage

```
tiledb_fragment_info(uri, ctx = tiledb_get_context())
```

Arguments

uri   A character variable with the URI of the array for which fragment info is re-
                  quested

ctx   tiledb_ctx object (optional)

Value

tiledb_fragment_info object

```
tiledb_fragment_info-class
```
An S4 class for a TileDB fragment info object

Description

An S4 class for a TileDB fragment info object

Slots

`ptr` An external pointer to the underlying implementation

```
tiledb_fragment_info_dense
```
Return if a fragment info index is dense

Description

Return if a fragment info index is dense

Usage

```
tiledb_fragment_info_dense(object, fid)
```

Arguments

`object`	A TileDB fragment info object
`fid`	A fragment object index

Value

A logical value indicating if the fragment is dense

`tiledb_fragment_info_dump`

Dump the fragment info to console

Description

Dump the fragment info to console

Usage

```
tiledb_fragment_info_dump(object)
```

Arguments

`object`	A TileDB fragment info object

Value

Nothing is returned, as a side effect the fragment info is displayed

---

`tiledb_fragment_info_get_cell_num`

Return a fragment info number of cells for a given fragment index

---

Description

Return a fragment info number of cells for a given fragment index

Usage

```
tiledb_fragment_info_get_cell_num(object, fid)
```

Arguments

`object`	A TileDB fragment info object
`fid`	A fragment object index

Value

A numeric value with the number of cells

tiledb_fragment_info_get_non_empty_domain_index

Return a fragment info non-empty domain from index

Description

TODO: Rework with type information

Usage

```
tiledb_fragment_info_get_non_empty_domain_index(object, fid, did, typestr)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index
did	A domain index
typestr	An optional character variable describing the data type which will be accessed from the schema if missing

Value

A TileDB Domain object

tiledb_fragment_info_get_non_empty_domain_name

Return a fragment info non-empty domain from name

Description

TODO: Rework with type information

Usage

```
tiledb_fragment_info_get_non_empty_domain_name(object, fid, dim_name, typestr)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index
dim_name	A character variable with the dimension name
typestr	An optional character variable describing the data type which will be accessed from the schema if missing

Value

A TileDB Domain object

---

tiledb_fragment_info_get_non_empty_domain_var_index
                    Return a fragment info non-empty domain variable from index

---

Description

Return a fragment info non-empty domain variable from index

Usage

```
tiledb_fragment_info_get_non_empty_domain_var_index(object, fid, did)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index
did	A domain index

Value

A character vector with two elements

---

tiledb_fragment_info_get_non_empty_domain_var_name
                    Return a fragment info non-empty domain variable from name

---

Description

Return a fragment info non-empty domain variable from name

Usage

```
tiledb_fragment_info_get_non_empty_domain_var_name(object, fid, dim_name)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index
dim_name	A character variable with the dimension name

Value

A character vector with two elements

---

tiledb_fragment_info_get_num

Return a fragment info number of fragments

Description

Return a fragment info number of fragments

Usage

```
tiledb_fragment_info_get_num(object)
```

Arguments

object   A TileDB fragment info object

Value

A numeric variable with the number of fragments

---

tiledb_fragment_info_get_size

Return a fragment info fragment size for a given fragment index

---

Description

Return a fragment info fragment size for a given fragment index

Usage

```
tiledb_fragment_info_get_size(object, fid)
```

Arguments

object   A TileDB fragment info object

fid   A fragment object index

Value

A numeric variable with the number of fragments

---

tiledb_fragment_info_get_timestamp_range

Return a fragment info timestamp range for a given fragment index

---

Description

Return a fragment info timestamp range for a given fragment index

Usage

```
tiledb_fragment_info_get_timestamp_range(object, fid)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index

Value

A Datetime vector with two elements for the range

---

tiledb_fragment_info_get_to_vacuum_num

Return the number of fragment info elements to be vacuumed

Description

Return the number of fragment info elements to be vacuumed

Usage

```
tiledb_fragment_info_get_to_vacuum_num(object)
```

Arguments

object	A TileDB fragment info object

Value

A numeric value with the number of to be vacuumed fragments

tiledb_fragment_info_get_to_vacuum_uri

Return fragment info URI of the to be vacuumed index

Description

Return fragment info URI of the to be vacuumed index

Usage

```
tiledb_fragment_info_get_to_vacuum_uri(object, fid)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index

Value

A character variable with the URI of the be vacuumed index

tiledb_fragment_info_get_unconsolidated_metadata_num

Return fragment info number of unconsolidated metadata

Description

Return fragment info number of unconsolidated metadata

Usage

```
tiledb_fragment_info_get_unconsolidated_metadata_num(object)
```

Arguments

object	A TileDB fragment info object

Value

A numeric value with the number of unconsolidated metadata

```
tiledb_fragment_info_get_version
```
Return a fragment info version for a given fragment index

Description

Return a fragment info version for a given fragment index

Usage

```
tiledb_fragment_info_get_version(object, fid)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index

Value

A integer value value with the version

```
tiledb_fragment_info_has_consolidated_metadata
```
Return if a fragment info index has consolidated metadata

Description

Return if a fragment info index has consolidated metadata

Usage

```
tiledb_fragment_info_has_consolidated_metadata(object, fid)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index

Value

A logical value indicating consolidated metadata

---

tiledb_fragment_info_sparse

Return if a fragment info index is sparse

Description

Return if a fragment info index is sparse

Usage

```
tiledb_fragment_info_sparse(object, fid)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index

Value

A logical value indicating if the fragment is sparse

---

tiledb_fragment_info_uri

Return a fragment info URI given its index

Description

Return a fragment info URI given its index

Usage

```
tiledb_fragment_info_uri(object, fid)
```

Arguments

object	A TileDB fragment info object
fid	A fragment object index

Value

A character variable with URI

`tiledb_get_all_metadata`

Return all TileDB Array Metadata objects as a named list

Description

Return all TileDB Array Metadata objects as a named list

Usage

```
tiledb_get_all_metadata(arr)
```

Arguments

arr   A TileDB Array object

Value

A named list with all Metadata objects indexed by the given key

`tiledb_get_context`   *Retrieve a TileDB context object from the package cache*

Description

Retrieve a TileDB context object from the package cache

Usage

```
tiledb_get_context()
```

Value

A TileDB context object

---

tiledb_get_metadata   *Return a TileDB Array Metadata object given by key*

Description

Return a TileDB Array Metadata object given by key

Usage

```
tiledb_get_metadata(arr, key)
```

Arguments

arr   A TileDB Array object

key   A character value describing a metadata key

Value

A object stored in the Metadata under the given key, or 'NULL' if none found.

---

tiledb_get_query_status
                 Retrieve the cached status of the last finalized query

---

Description

This function accesses the status of the last query without requiring the query object.

Usage

```
tiledb_get_query_status()
```

Value

The status of the last query

---

tiledb_get_vfs   *Retrieve a TileDB VFS object from the package environment and cache*

Description

Retrieve a TileDB VFS object from the package environment and cache

Usage

```
tiledb_get_vfs()
```

Value

A TileDB VFS object

---

tiledb_group   *Creates a 'tiledb_group' object*

Description

Creates a 'tiledb_group' object

Usage

```
tiledb_group(
  uri,
  type = c("READ", "WRITE"),
  ctx = tiledb_get_context(),
  cfg = NULL
)
```

Arguments

uri	Character variable with the URI of the new group object
type	Character variable with the query type value: one of "READ" or "WRITE"
ctx	(optional) A TileDB Context object; if not supplied the default context object is retrieved
cfg	(optional) A TileConfig object

Value

A 'group' object

tiledb_group-class *An S4 class for a TileDB Group object*

Description

An S4 class for a TileDB Group object

Slots

ptr An external pointer to the underlying implementation

tiledb_group_add_member

Add Member to TileDB Group

Description

Add Member to TileDB Group

Usage

```
tiledb_group_add_member(grp, uri, relative, name = NULL)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
uri	A character value with a new URI
relative	A logical value indicating whether URI is relative to the group
name	An optional character providing a name for the object, defaults to NULL

Value

The TileDB Group object, invisibly

tiledb_group_close   *Close a TileDB Group*

Description

Close a TileDB Group

Usage

```
tiledb_group_close(grp)
```

Arguments

grp   A TileDB Group object as for example returned by `tiledb_group()`

Value

The TileDB Group object but closed for reading or writing

tiledb_group_create   *Create a TileDB Group at the given path*

Description

Create a TileDB Group at the given path

Usage

```
tiledb_group_create(uri, ctx = tiledb_get_context())
```

Arguments

uri   Character variable with the URI of the new group
ctx   (optional) A TileDB Ctx object; if not supplied the default context object is
                       retrieved

Value

The uri path, invisibly

Examples

```
## Not run:
pth <- tempdir()
tiledb_group_create(pth)
tiledb_object_type(pth)

## End(Not run)
```

---

tiledb_group_delete *Deletes all written data from a 'tiledb_group' object*

Description

The group must be opened in 'MODIFY_EXCLUSIVE' mode, otherwise the function will error out.

Usage

```
tiledb_group_delete(grp, uri, recursive = FALSE)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
uri	Character variable with the URI of the group item to be deleted
recursive	A logical value indicating whether all data inside the group is to be deleted

Value

Nothing is returned, the function is invoked for the side-effect of group data removal.

---

tiledb_group_delete_metadata

Deletes Metadata from a TileDB Group

Description

Deletes Metadata from a TileDB Group

Usage

```
tiledb_group_delete_metadata(grp, key)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
key	A character value with they index under which the data will be written

Value

The TileDB Group object, invisibly

---

tiledb_group_get_all_metadata

Return all Metadata from a TileDB Group

Description

Return all Metadata from a TileDB Group

Usage

```
tiledb_group_get_all_metadata(grp)
```

Arguments

grp   A TileDB Group object as for example returned by tiledb_group()

Value

A named List with all Metadata objects index

---

tiledb_group_get_config

Get a TileDB Config from a TileDB Group

Description

Get a TileDB Config from a TileDB Group

Usage

```
tiledb_group_get_config(grp)
```

Arguments

grp   A TileDB Group object as for example returned by tiledb_group()

Value

The TileDB Config object of the TileDB Group object

tiledb_group_get_metadata
Accesses Metadata from a TileDB Group

Description

Accesses Metadata from a TileDB Group

Usage

```
tiledb_group_get_metadata(grp, key)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
key	A character value with the key of the metadata object to be retrieved

Value

The requested object, or NULL is not found

tiledb_group_get_metadata_from_index
Accesses Metadata by Index from a TileDB Group

Description

Accesses Metadata by Index from a TileDB Group

Usage

```
tiledb_group_get_metadata_from_index(grp, idx)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
idx	A numeric value with the index of the metadata object to be retrieved

Value

The requested object, or NULL is not found

`tiledb_group_has_metadata`

Checks for Metadata in a TileDB Group

Description

Checks for Metadata in a TileDB Group

Usage

```
tiledb_group_has_metadata(grp, key)
```

Arguments

grp   A TileDB Group object as for example returned by `tiledb_group()`

key   A character value with they index under which the data will be written

Value

A boolean value indicating with the object is present

`tiledb_group_is_open`   *Test if TileDB Group is open*

Description

Test if TileDB Group is open

Usage

```
tiledb_group_is_open(grp)
```

Arguments

grp   A TileDB Group object as for example returned by `tiledb_group()`

Value

A boolean indicating whether the TileDB Group object is open

`tiledb_group_is_relative`
Test if a Named Group is Using a Relative URI

Description

Test if a Named Group is Using a Relative URI

Usage

```
tiledb_group_is_relative(grp, name)
```

Arguments

grp	A TileDB Group object as for example returned by `tiledb_group()`
name	A character value with a group name

Value

A boolean indicating whether the group uses a relative URI or not

---

`tiledb_group_member` *Get a Member (Description) by Index from TileDB Group*

Description

This function returns a three-element character vector with the member object translated to character, uri, and optional name.

Usage

```
tiledb_group_member(grp, idx)
```

Arguments

grp	A TileDB Group object as for example returned by `tiledb_group()`
idx	A numeric value with the index of the metadata object to be retrieved

Value

A character vector with three elements: the member type, its uri, and name (or `""` if the member is unnamed).

`tiledb_group_member_count`

Get Member Count from TileDB Group

Description

Get Member Count from TileDB Group

Usage

```
tiledb_group_member_count(grp)
```

Arguments

`grp`	A TileDB Group object as for example returned by `tiledb_group()`

Value

The Count of Members in the TileDB Group object

`tiledb_group_member_dump`

Dump the TileDB Group to String

Description

Dump the TileDB Group to String

Usage

```
tiledb_group_member_dump(grp, recursive = FALSE)
```

Arguments

`grp`	A TileDB Group object as for example returned by `tiledb_group()`
`recursive`	A logical value indicating whether a recursive dump is desired, defaults to 'FALSE'. Note that recursive listings on remote object may be an expensive or slow operation.

Value

A character string

`tiledb_group_metadata_num`
Returns Number of Metadata Objects a TileDB Group

Description

Returns Number of Metadata Objects a TileDB Group

Usage

```
tiledb_group_metadata_num(grp)
```

Arguments

grp   A TileDB Group object as for example returned by `tiledb_group()`

Value

A numeric value with the number of metadata objects

`tiledb_group_open`   *Open a TileDB Group*

Description

Open a TileDB Group

Usage

```
tiledb_group_open(grp, type = c("READ", "WRITE", "MODIFY_EXCLUSIVE"))
```

Arguments

grp   A TileDB Group object as for example returned by `tiledb_group()`

type   A character value that must be either 'READ', 'WRITE' or 'MODIFY_EXCLUSIVE'

Value

The TileDB Group object but opened for reading or writing

---

tiledb_group_put_metadata

Write Metadata to a TileDB Group

Description

Write Metadata to a TileDB Group

Usage

```
tiledb_group_put_metadata(grp, key, val)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
key	A character value with they index under which the data will be written
val	An R object (numeric, int, or char vector) that will be stored

Value

On success boolean 'TRUE' is returned

---

tiledb_group_query_type

Return a TileDB Group query type

Description

Return a TileDB Group query type

Usage

```
tiledb_group_query_type(grp)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()

Value

A character value with the query type i.e. one of "READ" or "WRITE".

---

tiledb_group_remove_member

Remove Member from TileDB Group

Description

Remove Member from TileDB Group

Usage

```
tiledb_group_remove_member(grp, uri)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
uri	A character value with a the URI of the member to be removed, or (if added with a name) the name of the member

Value

The TileDB Group object, invisibly

---

tiledb_group_set_config

Set a TileDB Config for a TileDB Group

Description

Set a TileDB Config for a TileDB Group

Usage

```
tiledb_group_set_config(grp, cfg)
```

Arguments

grp	A TileDB Group object as for example returned by tiledb_group()
cfg	A TileDB Config object

Value

The TileDB Group object with added Config

---

tiledb_group_uri   *Return a TileDB Group URI*

Description

Return a TileDB Group URI

Usage

```
tiledb_group_uri(grp)
```

Arguments

grp   A TileDB Group object as for example returned by `tiledb_group()`

Value

A character value with the URI

---

tiledb_has_metadata   *Test if TileDB Array has Metadata*

Description

Test if TileDB Array has Metadata

Usage

```
tiledb_has_metadata(arr, key)
```

Arguments

arr   A TileDB Array object

key   A character value describing a metadata key

Value

A logical value indicating if the given key exists in the metdata of the given array

tiledb_is_supported_fs

Query if a TileDB backend is supported

Description

The scheme corresponds to the URI scheme for TileDB resouces.

Usage

```
tiledb_is_supported_fs(scheme, object = tiledb_get_context())
```

Arguments

scheme   URI string scheme ("file", "hdfs", "s3")

object   tiledb_ctx object

Details

Ex:

- `{file}:///path/to/file`
- `{hdfs}:///path/to/file`
- `{s3}://hostname:port/path/to/file`

Value

TRUE if tiledb backend is supported, FALSE otherwise

Examples

```
tiledb_is_supported_fs("file")
tiledb_is_supported_fs("s3")
```

tiledb_ndim,tiledb_array_schema-method

Return the number of dimensions associated with the
`tiledb_array_schema`

Description

Return the number of dimensions associated with the `tiledb_array_schema`

Usage

```
## S4 method for signature 'tiledb_array_schema'
tiledb_ndim(object)
```

Arguments

object   A TileDB Schema object

Value

integer number of dimensions

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(
  tiledb_attr("a1", type = "INT32"),
  tiledb_attr("a2", type = "FLOAT64")
))
tiledb_ndim(sch)
```

---

tiledb_ndim,tiledb_dim-method

Returns the number of dimensions for a tiledb domain object

Description

Returns the number of dimensions for a tiledb domain object

Usage

```
## S4 method for signature 'tiledb_dim'
tiledb_ndim(object)
```

Arguments

object   A `tiledb_dim` object

Value

An integer with the number of dimensions.

Examples

```
d1 <- tiledb_dim("d1", c(1L, 10L), 10L)
tiledb_ndim(d1)
```

tiledb_ndim,tiledb_domain-method

Returns the number of dimensions of the tiledb_domain

Description

Returns the number of dimensions of the tiledb_domain

Usage

```
## S4 method for signature 'tiledb_domain'
tiledb_ndim(object)
```

Arguments

object	tiledb_domain

Value

integer number of dimensions

Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64")))
tiledb_ndim(dom)
dom <- tiledb_domain(dims = c(
  tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64"),
  tiledb_dim("d2", c(0.5, 100.0), type = "FLOAT64")
))
tiledb_ndim(dom)
```

---

tiledb_ndrectangle *Creates a* tiledb_ndrectangle *object*

Description

Creates a tiledb_ndrectangle object

Usage

```
tiledb_ndrectangle(dom, ctx = tiledb_get_context())
```

Arguments

dom	A TileDB Domain object for which the NDRectangle object is created
ctx	(optional) A TileDB Ctx object

Value

The `tiledb_ndrectangle` object

Examples

```
if (tiledb_version(TRUE) >= "2.25.0") {
  dom <- tiledb_domain(dim = tiledb_dim("d1", c(1L, 100L), type = "INT32"))
  ndr <- tiledb_ndrectangle(dom)
}
```

---

tiledb_ndrectangle-class
                        An S4 class for a TileDB NDRectangle object

---

Description

An S4 class for a TileDB NDRectangle object

Slots

`ptr` An external pointer to the underlying NDRectangle object

---

tiledb_ndrectangle_datatype
                        Get the datatype of a named tiledb_ndrectangle *dimension*

Description

Get the datatype of a named `tiledb_ndrectangle` dimension

Usage

```
tiledb_ndrectangle_datatype(ndr, dimname)
```

Arguments

ndr	A TileDB NDRectangle object
dimname	A character variable with the dimension for which to get a datatype

Value

The `tiledb_ndrectangle` dimension datatype as a character

Examples

```
if (tiledb_version(TRUE) >= "2.26.0") {
  dom <- tiledb_domain(dim = tiledb_dim("d1", c(1L, 100L), type = "INT32"))
  ndr <- tiledb_ndrectangle(dom)
  tiledb_ndrectangle_datatype(ndr, "d1")
}
```

---

tiledb_ndrectangle_datatype_by_ind

Get the datatype of a tiledb_ndrectangle *dimension by index*

Description

Get the datatype of a tiledb_ndrectangle dimension by index

Usage

```
tiledb_ndrectangle_datatype_by_ind(ndr, dim)
```

Arguments

ndr	A TileDB NDRectangle object
dim	Am integer value for the dimension for which to get a datatype

Value

The tiledb_ndrectangle dimension datatype as a character

Examples

```
if (tiledb_version(TRUE) >= "2.26.0") {
  dom <- tiledb_domain(dim = tiledb_dim("d1", c(1L, 100L), type = "INT32"))
  ndr <- tiledb_ndrectangle(dom)
  tiledb_ndrectangle_datatype_by_ind(ndr, 0)
}
```

---

tiledb_ndrectangle_dim_num

Get the number of dimensions for tiledb_ndrectangle *object*

Description

Get the number of dimensions for tiledb_ndrectangle object

Usage

```
tiledb_ndrectangle_dim_num(ndr)
```

Arguments

ndr   A TileDB NDRectangle object

Value

The number of dimentiones for the tiledb_ndrectangle

Examples

```
if (tiledb_version(TRUE) >= "2.26.0") {
  dom <- tiledb_domain(dim = tiledb_dim("d1", c(1L, 100L), type = "INT32"))
  ndr <- tiledb_ndrectangle(dom)
  tiledb_ndrectangle_dim_num(ndr)
}
```

---

tiledb_ndrectangle_get_range

Get a range from a tiledb_ndrectangle *object*

Description

Get a range from a tiledb_ndrectangle object

Usage

```
tiledb_ndrectangle_get_range(ndr, dimname)
```

Arguments

ndr   A TileDB NDRectangle object

dimname   A character variable with the dimension for which to get a range

Value

The `tiledb_ndrectangle` range as a two-element vector

Examples

```
if (tiledb_version(TRUE) >= "2.26.0") {
  dom <- tiledb_domain(dim = tiledb_dim("d1", c(1L, 100L), type = "INT32"))
  ndr <- tiledb_ndrectangle(dom)
  ndr <- tiledb_ndrectangle_set_range(ndr, "d1", 50, 500)
  tiledb_ndrectangle_get_range(ndr, "d1")
}
```

---

tiledb_ndrectangle_set_range

Set a range on a `tiledb_ndrectangle` *object*

Description

Set a range on a `tiledb_ndrectangle` object

Usage

```
tiledb_ndrectangle_set_range(ndr, dimname, start, end)
```

Arguments

ndr	A TileDB NDRectangle object
dimname	A character variable with the dimension for which to set a range
start	The lower end of the range to be set
end	The upper end of the range to be set

Value

The modified `tiledb_ndrectangle` object

Start and end values have to be of the same data type as the type of the selected dimension. The set of allowed type includes the different integer types as well as string dimensions.

Examples

```
if (tiledb_version(TRUE) >= "2.26.0") {
  dom <- tiledb_domain(dim = tiledb_dim("d1", c(1L, 100L), type = "INT32"))
  ndr <- tiledb_ndrectangle(dom)
  ndr <- tiledb_ndrectangle_set_range(ndr, "d1", 50, 500)
}
```

---

tiledb_num_metadata   *Return count of TileDB Array Metadata objects*

Description

Return count of TileDB Array Metadata objects

Usage

```
tiledb_num_metadata(arr)
```

Arguments

arr   A TileDB Array object

Value

A integer variable with the number of Metadata objects

---

tiledb_object_ls   *List TileDB resources at a given root URI path*

Description

List TileDB resources at a given root URI path

Usage

```
tiledb_object_ls(uri, filter = NULL, ctx = tiledb_get_context())
```

Arguments

uri   uri path to walk
filter   optional filtering argument, default is "NULL", currently unused
ctx   tiledb_ctx object (optional)

Value

a dataframe with object type, object uri string columns

`tiledb_object_mv`	*Move a TileDB resource to new uri path*

Description

Raises an error if either uri is invalid, or the old uri resource is not a tiledb object

Usage

```
tiledb_object_mv(old_uri, new_uri, ctx = tiledb_get_context())
```

Arguments

`old_uri`	old uri of existing tiledb resource
`new_uri`	new uri to move tiledb resource
`ctx`	tiledb_ctx object (optional)

Value

new uri of moved tiledb resource

`tiledb_object_rm`	*Removes a TileDB resource*

Description

Raises an error if the uri is invalid, or the uri resource is not a tiledb object

Usage

```
tiledb_object_rm(uri, ctx = tiledb_get_context())
```

Arguments

`uri`	path to TileDB resource
`ctx`	tiledb_ctx object (optional)

Value

uri of removed TileDB resource

---

tiledb_object_type   *Return the TileDB object type string of a TileDB resource*

Description

Object types:

- "ARRAY", dense or sparse TileDB array
- "GROUP", TileDB group
- '"INVALID"'", not a TileDB resource

Usage

```
tiledb_object_type(uri, ctx = tiledb_get_context())
```

Arguments

uri	path to TileDB resource
ctx	tiledb_ctx object (optional)

Value

TileDB object type string

---

tiledb_object_walk   *Recursively discover TileDB resources at a given root URI path*

Description

Recursively discover TileDB resources at a given root URI path

Usage

```
tiledb_object_walk(
  uri,
  order = c("PREORDER", "POSTORDER"),
  ctx = tiledb_get_context()
)
```

Arguments

uri	root uri path to walk
order	traversal order, one of "PREORDER" and "POSTORDER" (default "PREORDER")
ctx	tiledb_ctx object (optional)

Value

a dataframe with object type, object uri string columns

---

tiledb_put_metadata *Store an object in TileDB Array Metadata under given key*

Description

Store an object in TileDB Array Metadata under given key

Usage

```
tiledb_put_metadata(arr, key, val)
```

Arguments

arr	A TileDB Array object
key	A character value describing a metadata key
val	An object to be stored

Value

A boolean value indicating success

---

tiledb_query *Creates a 'tiledb_query' object*

Description

Creates a 'tiledb_query' object

Usage

```
tiledb_query(
  array,
  type = if (tiledb_version(TRUE) >= "2.12.0") {
      c("READ", "WRITE", "DELETE",
    "MODIFY_EXCLUSIVE")
 } else {
      c("READ", "WRITE")
  },
  ctx = tiledb_get_context()
)
```

Arguments

`array`	A TileDB Array object
`type`	A character value that must be one of 'READ', 'WRITE', or 'DELETE' (for TileDB >= 2.12.0)
`ctx`	(optional) A TileDB Ctx object

Value

'tiledb_query' object

`tiledb_query-class`   *An S4 class for a TileDB Query object*

Description

An S4 class for a TileDB Query object

Slots

`ptr` An external pointer to the underlying implementation

`tiledb_query_add_range`

                              Set a range for a given query

Description

Set a range for a given query

Usage

```
tiledb_query_add_range(query, schema, attr, lowval, highval, stride = NULL)
```

Arguments

`query`	A TileDB Query object
`schema`	A TileDB Schema object
`attr`	An character variable with a dimension name for which the range is set
`lowval`	The lower value of the range to be set
`highval`	The higher value of the range to be set
`stride`	An optional stride value for the range to be set

Value

The query object, invisibly

tiledb_query_add_range_with_type

Set a range for a given query, also supplying type

Description

Set a range for a given query, also supplying type

Usage

```
tiledb_query_add_range_with_type(
  query,
  idx,
  datatype,
  lowval,
  highval,
  stride = NULL
)
```

Arguments

query	A TileDB Query object
idx	An integer index, zero based, of the dimensions
datatype	A character value containing the data type
lowval	The lower value of the range to be set
highval	The highre value of the range to be set
stride	An optional stride value for the range to be set

Value

The query object, invisibly

tiledb_query_alloc_buffer_ptr_char

Allocate a Query buffer for reading a character attribute

Description

Allocate a Query buffer for reading a character attribute

Usage

```
tiledb_query_alloc_buffer_ptr_char(sizeoffsets, sizedata, nullable = FALSE)
```

Arguments

`sizeoffsets`	A numeric value with the size of the offsets vector
`sizedata`	A numeric value of the size of the data string
`nullable`	An optional boolean indicating whether the column can have NULLs

Value

An external pointer to the allocated buffer object

`tiledb_query_apply_aggregate`

Run an aggregate oprtation on the given query attribute

Description

Run an aggregate oprtation on the given query attribute

Usage

```
tiledb_query_apply_aggregate(
  query,
  attrname,
  operation = c("Count", "NullCount", "Min", "Max", "Mean", "Sum"),
  nullable = TRUE
)
```

Arguments

`query`	A TileDB Query object
`attrname`	The name of an attribute
`operation`	The name of aggregation operation
`nullable`	A boolean toggle whether the attribute is nullable

Value

The value of the aggregation

`tiledb_query_buffer_alloc_ptr`

Allocate a Query buffer for a given type

Description

This function allocates a query buffer for the given data type.

Usage

```
tiledb_query_buffer_alloc_ptr(
  query,
  datatype,
  ncells,
  nullable = FALSE,
  varnum = 1
)
```

Arguments

query	A TileDB Query object
datatype	A character value containing the data type
ncells	A number of elements (not bytes)
nullable	Optional boolean parameter indicating whether missing values are allowed (for which another column is allocated), default is FALSE
varnum	Option intgeter parameter for the number of elemements per variable, default is one

Value

An external pointer to the allocated buffer object

`tiledb_query_condition`

Creates a 'tiledb_query_condition' object

Description

Creates a 'tiledb_query_condition' object

Usage

```
tiledb_query_condition(ctx = tiledb_get_context())
```

Arguments

ctx   (optional) A TileDB Ctx object; if not supplied the default context object is
                   retrieved

Value

A 'tiledb_query_condition' object

---

tiledb_query_condition-class

An S4 class for a TileDB QueryCondition object

Description

An S4 class for a TileDB QueryCondition object

Slots

ptr An external pointer to the underlying implementation

init A logical variable tracking if the query condition object has been initialized

---

tiledb_query_condition_combine

Combine two 'tiledb_query_condition' objects

Description

Combines two query condition object using a relatiional operator. Support for operator 'AND' is
generally available, the 'OR' operator is available if TileDB 2.10 or newer is used.

Usage

```
tiledb_query_condition_combine(lhs, rhs, op)
```

Arguments

lhs   A 'tiledb_query_condition' object on the left-hand side of the relation

rhs   A 'tiledb_query_condition' object on the left-hand side of the relation

op   A character value with then relation, this must be one of 'AND', 'OR' or 'NOT'.

Value

The combined 'tiledb_query_condition' object

`tiledb_query_condition_create`

Create a query condition for vector 'IN' and 'NOT_IN' operations

---

Description

Uses 'IN' and 'NOT_IN' operators on given attribute

Usage

```
tiledb_query_condition_create(
  name,
  values,
  op = "IN",
  ctx = tiledb_get_context()
)
```

Arguments

`name`	A character value with the scheme attribute name
`values`	A vector wiith the given values, supported types are integer, double, integer64 and charactor
`op`	(optional) A character value with the chosen set operation, this must be one of 'IN' or 'NOT_IN'; default to 'IN'
`ctx`	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

A query condition object is returned

`tiledb_query_condition_init`

Initialize a 'tiledb_query_condition' object

Description

Initializes (and possibly allocates) a query condition object using a triplet of attribute name, comparison value, and operator. Six types of conditions are supported, they all take a single scalar comparison argument and attribute to compare against. At present only integer or numeric attribute comparisons are implemented.

Usage

```
tiledb_query_condition_init(
  attr,
  value,
  dtype,
  op,
  qc = tiledb_query_condition()
)
```

Arguments

attr	A character value with the scheme attribute name
value	A scalar value that the attribute is compared against
dtype	A character value with the TileDB data type of the attribute column, for example 'FLOAT64' or 'INT32'
op	A character value with then comparison operation, this must be one of 'LT', 'LE', 'GT', 'GE', 'EQ', 'NE'.
qc	(optional) A 'tiledb_query_condition' object to be initialized by this call, if none is given a new one is allocated.

Value

The initialized 'tiledb_query_condition' object

---

tiledb_query_condition_set_use_enumeration
Enable use of enumeration in query condition

Description

Set a boolean toggle to signal use of enumeration in query condtion (TileDB 2.17 or later)

Usage

```
tiledb_query_condition_set_use_enumeration(
  qc,
  use_enum,
  ctx = tiledb_get_context()
)
```

Arguments

qc	A 'tiledb_query_condition' object
use_enum	A boolean to set (if TRUE) or unset (if FALSE) enumeration use
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

Value

Nothing is retuned, the function is invoked for the side effect

---

tiledb_query_create_buffer_ptr

Allocate and populate a Query buffer for a given object of a given data type.

Description

This function allocates a query buffer for the given data object of the given type and assigns the object content to the buffer.

Usage

```
tiledb_query_create_buffer_ptr(query, datatype, object)
```

Arguments

query	A TileDB Query object
datatype	A character value containing the data type
object	A vector object of the given type

Value

An external pointer to the allocated buffer object

---

tiledb_query_create_buffer_ptr_char

Allocate and populate a Query buffer for writing the given char vector

---

Description

Allocate and populate a Query buffer for writing the given char vector

Usage

```
tiledb_query_create_buffer_ptr_char(query, varvec)
```

Arguments

query	A TileDB Query object
varvec	A vector of strings

Value

An external pointer to the allocated buffer object

---

tiledb_query_ctx   *Return query context object*

Description

Return query context object

Usage

```
tiledb_query_ctx(query)
```

Arguments

query   A TileDB Query object

Value

A TileDB Context object retrieved from the query

---

tiledb_query_export_buffer
                              Export Query Buffer to Pair of Arrow IO Pointers

---

Description

This function exports the named buffer from a 'READ' query to two Arrow C pointers.

Usage

```
tiledb_query_export_buffer(query, name, ctx = tiledb_get_context())
```

Arguments

query   A TileDB Query object

name   A character variable identifying the buffer

ctx   tiledb_ctx object (optional)

Value

A nanoarrow object (which is an external pointer to an Arrow Array with the Arrow Schema stored as the external pointer tag) classed as an S3 object

---

`tiledb_query_finalize` *Finalize TileDB Query*

Description

Finalize TileDB Query

Usage

```
tiledb_query_finalize(query)
```

Arguments

query   A TileDB Query object

Value

A character value, either 'READ' or 'WRITE'

`tiledb_query_get_buffer_char`
                              Retrieve content from a Query character buffer

---

Description

This function uses a query buffer for a character attribute or dimension and returns its content.

Usage

```
tiledb_query_get_buffer_char(bufptr, sizeoffsets = 0, sizestring = 0)
```

Arguments

bufptr   An external pointer with a query buffer
sizeoffsets   An optional argument for the length of the internal offsets vector
sizestring   An optional argument for the length of the internal string

Value

An R object as resulting from the query

`tiledb_query_get_buffer_ptr`
                        Retrieve content from a Query buffer

Description

This function uses a query buffer and returns its content.

Usage

```
tiledb_query_get_buffer_ptr(bufptr)
```

Arguments

bufptr   An external pointer with a query buffer

Value

An R object as resulting from the query

`tiledb_query_get_est_result_size`
                        Retrieve the estimated result size for a query and attribute

---

Description

When reading from sparse arrays, one cannot know beforehand how big the result will be (unless
one actually executes the query). This function offers a way to get the estimated result size for the
given attribute. As TileDB does not actually execute the query, getting the estimated result is very
fast.

Usage

```
tiledb_query_get_est_result_size(query, name)
```

Arguments

query   A TileDB Query object

name   A variable with an attribute name

Value

An estimate of the query result size

`tiledb_query_get_est_result_size_var`

Retrieve the estimated result size for a query and variable-sized attribute

---

Description

When reading variable-length attributes from either dense or sparse arrays, one cannot know beforehand how big the result will be (unless one actually executes the query). This function offers a way to get the estimated result size for the given attribute. As TileDB does not actually execute the query, getting the estimated result is very fast.

Usage

```
tiledb_query_get_est_result_size_var(query, name)
```

Arguments

query	A TileDB Query object
name	A variable with an attribute name

Value

An estimate of the query result size

`tiledb_query_get_fragment_num`

Retrieve the Number of Fragments for Query

Description

This function is only applicable to 'WRITE' queries.

Usage

```
tiledb_query_get_fragment_num(query)
```

Arguments

query	A TileDB Query object

Value

An integer with the number of fragments for the given query

`tiledb_query_get_fragment_timestamp_range`

Retrieve the timestamp range for a given Query Fragment

Description

This function is only applicable to 'WRITE' queries. The time resolution in TileDB is millseconds since the epoch so an R `Datetime` vector is returned.

Usage

```
tiledb_query_get_fragment_timestamp_range(query, idx)
```

Arguments

query	A TileDB Query object
idx	An integer (or numeric) index ranging from zero to the number of fragments minus 1

Value

A two-element datetime vector with the start and end time of the fragment write.

`tiledb_query_get_fragment_uri`

Retrieve the URI for a given Query Fragment

Description

This function is only applicable to 'WRITE' queries.

Usage

```
tiledb_query_get_fragment_uri(query, idx)
```

Arguments

query	A TileDB Query object
idx	An integer (or numeric) index ranging from zero to the number of fragments minus 1

Value

An character value with the fragment URI

`tiledb_query_get_layout`

Get TileDB Query layout

Description

Get TileDB Query layout

Usage

```
tiledb_query_get_layout(query)
```

Arguments

query   A TileDB Query object

Value

The TileDB Query layout as a string

`tiledb_query_get_range`

Retrieve the query range for a query dimension and range index

---

Description

Retrieve the query range for a query dimension and range index

Usage

```
tiledb_query_get_range(query, dimidx, rngidx)
```

Arguments

query   A TileDB Query object

dimidx   An integer or numeric index selecting the dimension

rngidx   An integer or numeric index selection the given range for the dimension

Value

An integer vector with elements start, end and stride for the query range for the given dimension and range index

---

`tiledb_query_get_range_num`

Retrieve the number of ranges for a query dimension

Description

Retrieve the number of ranges for a query dimension

Usage

```
tiledb_query_get_range_num(query, idx)
```

Arguments

query	A TileDB Query object
idx	An integer or numeric index selecting the dimension

Value

An integer with the number of query range for the given dimensions

---

`tiledb_query_get_range_var`

Retrieve the query range for a variable-sized query dimension and range index

---

Description

Retrieve the query range for a variable-sized query dimension and range index

Usage

```
tiledb_query_get_range_var(query, dimidx, rngidx)
```

Arguments

query	A TileDB Query object
dimidx	An integer index selecting the variable-sized dimension
rngidx	An integer index selection the given range for the dimension

Value

An string vector with elements start and end for the query range for the given dimension and range index

---

```
tiledb_query_import_buffer
```
Import to Query Buffer from Pair of Arrow IO Pointers

Description

This function imports to the named buffer for a 'WRITE' query from two Arrow exerternal pointers.

Usage

```
tiledb_query_import_buffer(
  query,
  name,
  nanoarrowptr,
  ctx = tiledb_get_context()
)
```

Arguments

query	A TileDB Query object
name	A character variable identifying the buffer
nanoarrowptr	A nanoarrow object (which is an external pointer to an Arrow Array with the Arrow Schema stored as the external pointer tag) classed as an S3 object
ctx	tiledb_ctx object (optional)

Value

The update Query external pointer is returned

```
tiledb_query_result_buffer_elements
```
Get TileDB Query result buffer element size

Description

The underlying library functions returns a pair of values as a vector of length two. The first number is the number of element offsets for variable size attributes (and always zero for fixed-sized attributes and coordinates). The second is the number of elements in the data buffer. For variable-sized attributes the first number is the number of cells read (and hence the number of offsets), the second number is the number of elements in the data buffer.

Usage

```
tiledb_query_result_buffer_elements(query, attr)
```

Arguments

query	A TileDB Query object
attr	A character value containing the attribute

Details

As this function was first made available when only a scalar (corresponding to the second result) was returned, we still return that value.

Value

A integer with the number of elements in the results buffer for the given attribute

See Also

tiledb_query_result_buffer_elements_vec

---

tiledb_query_result_buffer_elements_vec
Get TileDB Query result buffer element size pair as vector

Description

The underlying library functions returns a pair of values as a vector of length two. The first number is the number of element offsets for variable size attributes (and always zero for fixed-sized attributes and coordinates). The second is the number of elements in the data buffer. For variable-sized attributes the first number is the number of cells read (and hence the number of offsets), the second number is the number of elements in the data buffer. In the case of a nullable attribute, a third element is returned with the size of the validity buffer.

Usage

```
tiledb_query_result_buffer_elements_vec(query, attr, nullable = FALSE)
```

Arguments

query	A TileDB Query object
attr	A character value containing the attribute
nullable	A logical variable that is 'TRUE' to signal that the attribute is nullable, and 'FALSE' otherwise

Value

A vector with the number of elements in the offsets buffer (and zero for fixed-size attribute or dimensions), the number elements in the results buffer for the given attribute, and (if nullable) a third element with the validity buffer size.

See Also

tiledb_query_result_buffer_elements

---

tiledb_query_set_buffer

Set TileDB Query buffer

Description

This function allocates query buffers directly from R vectors in case the types match: `integer`, `double`, `logical`. For more general types see `tiledb_query_buffer_alloc_ptr` and `tiledb_query_buffer_assign_ptr`

Usage

```
tiledb_query_set_buffer(query, attr, buffer)
```

Arguments

query	A TileDB Query object
attr	A character value containing the attribute
buffer	A vector providing the query buffer

Value

The modified query object, invisisibly

---

tiledb_query_set_buffer_ptr

Assigns to a Query buffer for a given attribute

Description

This function assigns a given query buffer to a query.

Usage

```
tiledb_query_set_buffer_ptr(query, attr, bufptr)
```

Arguments

query	A TileDB Query object
attr	A character value containing the attribute
bufptr	An external pointer with a query buffer

Value

The modified query object, invisibly

`tiledb_query_set_buffer_ptr_char`

Assign a buffer to a Query attribute

Description

Assign a buffer to a Query attribute

Usage

```
tiledb_query_set_buffer_ptr_char(query, attr, bufptr)
```

Arguments

query	A TileDB Query object
attr	A character value containing the attribute
bufptr	An external pointer with a query buffer

Value

The modified query object, invisibly

`tiledb_query_set_condition`

Set a query combination object for a query

Description

Set a query combination object for a query

Usage

```
tiledb_query_set_condition(query, qc)
```

Arguments

query	A TileDB Query object
qc	A TileDB Query Combination object

Value

The modified query object, invisibly

---

tiledb_query_set_layout

Set TileDB Query layout

Description

Set TileDB Query layout

Usage

```
tiledb_query_set_layout(
  query,
  layout = c("COL_MAJOR", "ROW_MAJOR", "GLOBAL_ORDER", "UNORDERED")
)
```

Arguments

query	A TileDB Query object
layout	A character variable with the layout; must be one of "COL_MAJOR", "ROW_MAJOR", "GLOBAL_ORDER", "UNORDERED")

Value

The modified query object, invisibly

---

tiledb_query_set_subarray

Set subarray for TileDB Query object

Description

Set subarray for TileDB Query object

Usage

```
tiledb_query_set_subarray(query, subarray, type)
```

Arguments

query	A TileDB Query object
subarray	A subarry vector object
type	An optional type as a character, if missing type is inferred from the vector.

Value

The modified query object, invisibly

---

tiledb_query_stats   *Return query statistics as a JSON string*

Description

Return query statistics as a JSON string

Usage

```
tiledb_query_stats(query)
```

Arguments

query   A TileDB Query object

Value

A JSON-formatted string with context statistics

---

tiledb_query_status   *Get TileDB Query status*

Description

Get TileDB Query status

Usage

```
tiledb_query_status(query)
```

Arguments

query   A TileDB Query object

Value

A character value describing the query status

tiledb_query_submit   *Submit TileDB Query*

Description

Note that the query object may need to be finalized via `tiledb_query_finalize`.

Usage

```
tiledb_query_submit(query)
```

Arguments

query   A TileDB Query object

Value

The modified query object, invisibly

tiledb_query_submit_async

                            Submit TileDB Query asynchronously without a callback returning immediately

Description

Note that the query object may need to be finalized via `tiledb_query_finalize`.

Usage

```
tiledb_query_submit_async(query)
```

Arguments

query   A TileDB Query object

Value

The modified query object, invisibly

---

tiledb_query_type   *Return TileDB Query type*

Description

Return TileDB Query type

Usage

```
tiledb_query_type(query)
```

Arguments

query   A TileDB Query object

Value

A character value, either 'READ' or 'WRITE'

---

tiledb_schema_get_dim_attr_status
                            Get Dimension or Attribute Status

---

Description

Note that this function is an unexported internal function that can be called using the colons as in
tiledb:::tiledb_schema_get_dim_attr_status(sch).

Usage

```
tiledb_schema_get_dim_attr_status(sch)
```

Arguments

sch   A TileDB Schema object

Value

An integer vector where each element corresponds to a schema entry, and a value of one signals
dimension and a value of two an attribute.

tiledb_schema_get_enumeration_status
Get Dimension or Attribute Status

Description

Note that this function is an unexported internal function that can be called using the colons as in
`tiledb:::tiledb_schema_get_enumeration_status(sch)`.

Usage

```
tiledb_schema_get_enumeration_status(sch)
```

Arguments

sch   A TileDB Schema object

Value

An integer vector where each element corresponds to a schema entry, and a value of one signals
dimension and a value of two an attribute.

tiledb_schema_get_names
Get all Dimension and Attribute Names

Description

Get all Dimension and Attribute Names

Usage

```
tiledb_schema_get_names(sch)
```

Arguments

sch   A TileDB Schema object

Value

A character vector of dimension and attribute names

---

tiledb_schema_get_types
Get all Dimension and Attribute Types

Description

Get all Dimension and Attribute Types

Usage

```
tiledb_schema_get_types(sch)
```

Arguments

sch   A TileDB Schema object

Value

A character vector of dimension and attribute data types

---

tiledb_schema_object   *Succinctly describe a TileDB array schema*

Description

This is an internal function that is not exported.

Usage

```
tiledb_schema_object(array)
```

Arguments

array   A TileDB Array object

Value

A list containing two data frames, one describing the overall array as well as one with descriptions
about dimensions and attributes in the schema

---

tiledb_set_context   *Store a TileDB context object in the package cache*

Description

Store a TileDB context object in the package cache

Usage

```
tiledb_set_context(ctx)
```

Arguments

ctx   A TileDB context object

Value

NULL, invisibly. The function is invoked for the side-effect of storing the VFS object.

---

tiledb_set_vfs   *Store a TileDB VFS object in the package environment*

Description

Store a TileDB VFS object in the package environment

Usage

```
tiledb_set_vfs(vfs)
```

Arguments

vfs   A TileDB VFS object

Value

NULL, invisibly. The function is invoked for the side-effect of storing the VFS object.

---

tiledb_stats_disable   *Disable internal TileDB statistics counters*

Description

This function ends the collection of internal statistics.

Usage

```
tiledb_stats_disable()
```

---

tiledb_stats_dump   *Dumps internal TileDB statistics to file or stdout*

Description

Dumps internal TileDB statistics to file or stdout

Usage

```
tiledb_stats_dump(path)
```

Arguments

path   Character variable with path to stats file; if the empty string is passed then the
              result is displayed on stdout.

Examples

```
pth <- tempfile()
tiledb_stats_dump(pth)
cat(readLines(pth)[1:10], sep = "\n")
```

---

tiledb_stats_enable   *Enable internal TileDB statistics counters*

Description

This function starts the collection of internal statistics.

Usage

```
tiledb_stats_enable()
```

---

tiledb_stats_print   *Print internal TileDB statistics*

Description

This function is a convenience wrapper for `tiledb_stats_dump`.

Usage

```
tiledb_stats_print()
```

---

tiledb_stats_raw_dump   *Dumps internal TileDB statistics as JSON to a string*

Description

This function requires TileDB Embedded 2.0.3 or later.

Usage

```
tiledb_stats_raw_dump()
```

Examples

```
txt <- tiledb_stats_raw_dump()
cat(txt, "\n")
```

---

tiledb_stats_raw_get   *Gets internal TileDB statistics as JSON string*

Description

This function is a (now deprecated) convenience wrapper for `tiledb_stats_raw_dump` and returns
the result as a JSON string. It required TileDB Embedded 2.0.3 or later.

Usage

```
tiledb_stats_raw_get()
```

`tiledb_stats_raw_print`
Print internal TileDB statistics as JSON

Description

This function is a convenience wrapper for `tiledb_stats_raw_dump`. It required TileDB Embedded 2.0.3 or later.

Usage

```
tiledb_stats_raw_print()
```

`tiledb_stats_reset`   *Reset internal TileDB statistics counters*

Description

This function resets the counters for internal statistics.

Usage

```
tiledb_stats_reset()
```

`tiledb_subarray`   *Constructs a* `tiledb_subarray` *object from a TileDB Query*

Description

Constructs a `tiledb_subarray` object from a TileDB Query

Usage

```
tiledb_subarray(query)
```

Arguments

query   A TileDB Query Object

Value

tiledb_subarray object

`tiledb_subarray-class` *An S4 class for a TileDB Subarray*

Description

An S4 class for a TileDB Subarray

Slots

`ptr` External pointer to the underlying implementation

`tiledb_subarray_to_query`

Apply a Subarray to a Query

Description

Apply a Subarray to a Query

Usage

```
tiledb_subarray_to_query(query, subarray)
```

Arguments

query	A TileDB Query Object
subarray	A TileDB Subarray Object

Value

tiledb_query object

---

tiledb_version   *The version of the libtiledb library*

Description

The version of the libtiledb library

Usage

```
tiledb_version(compact = FALSE)
```

Arguments

compact   Logical value indicating wheter a compact `package_version` object should be
                 returned

Value

An named int vector c(major, minor, patch), or if select, a `package_version` object

Examples

```
tiledb_version()
tiledb_version(compact = TRUE)
```

---

tiledb_vfs   *Creates a* tiledb_vfs *object*

Description

Creates a `tiledb_vfs` object

Usage

```
tiledb_vfs(config = NULL, ctx = tiledb_get_context())
```

Arguments

config   (optional) character vector of config parameter names, values
ctx   (optional) A TileDB Ctx object

Value

The `tiledb_vfs` object

Examples

```
# default configuration
vfs <- tiledb_vfs()
```

---

tiledb_vfs-class   *An S4 class for a TileDB VFS object*

Description

An S4 class for a TileDB VFS object

Slots

ptr An external pointer to the underlying implementation

---

tiledb_vfs_close   *Close a TileDB VFS Filehandle*

Description

Close a TileDB VFS Filehandle

Usage

```
tiledb_vfs_close(fh, ctx = tiledb_get_context())
```

Arguments

fh	A TileDB VFS Filehandle external pointer as returned from `tiledb_vfs_open`
ctx	(optional) A TileDB Ctx object

Value

The result of the close operation is returned.

---

tiledb_vfs_copy_dir   *Copy a directory recursively to VFS*

Description

Copy a directory recursively to VFS

Usage

```
tiledb_vfs_copy_dir(dir, uri, vfs = tiledb_get_vfs())
```

Arguments

dir	Character variable with a local directory path
uri	Character variable with a URI describing a directory path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value of the copied directory

---

tiledb_vfs_copy_file   *Copy a file to VFS*

Description

Copy a file to VFS

Usage

```
tiledb_vfs_copy_file(file, uri, vfs = tiledb_get_vfs())
```

Arguments

file	Character variable with a local file path
uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value of the copied file

`tiledb_vfs_create_bucket`
Create a VFS Bucket

Description

Create a VFS Bucket

Usage

```
tiledb_vfs_create_bucket(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value

`tiledb_vfs_create_dir` *Create a VFS Directory*

Description

Create a VFS Directory

Usage

```
tiledb_vfs_create_dir(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a diretory path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value of the created directory

`tiledb_vfs_dir_size`   *Return VFS Directory Size*

Description

Return VFS Directory Size

Usage

```
tiledb_vfs_dir_size(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The size of the directory

`tiledb_vfs_empty_bucket`

Empty a VFS Bucket

Description

Empty a VFS Bucket

Usage

```
tiledb_vfs_empty_bucket(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

Value

The URI value that was emptied

tiledb_vfs_file_size   *Return VFS File Size*

Description

Return VFS File Size

Usage

```
tiledb_vfs_file_size(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The size of the file

tiledb_vfs_is_bucket   *Check for VFS Bucket*

Description

Check for VFS Bucket

Usage

```
tiledb_vfs_is_bucket(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

Value

A boolean value indicating if it is a valid bucket

Examples

```
## Not run:
cfg <- tiledb_config()
cfg["vfs.s3.region"] <- "us-west-1"
ctx <- tiledb_ctx(cfg)
vfs <- tiledb_vfs()
tiledb_vfs_is_bucket(vfs, "s3://tiledb-public-us-west-1/test-array-4x4")

## End(Not run)
```

---

tiledb_vfs_is_dir   *Test for VFS Directory*

Description

Test for VFS Directory

Usage

```
tiledb_vfs_is_dir(uri, vfs = tiledb_get_vfs())
```

Arguments

uri   Character variable with a URI describing a diretory path

vfs   A TileDB VFS object; default is to use a cached value.

Value

A boolean value indicating if it is a directory

---

tiledb_vfs_is_empty_bucket
                           Check for empty VFS Bucket

Description

Check for empty VFS Bucket

Usage

```
tiledb_vfs_is_empty_bucket(uri, vfs = tiledb_get_vfs())
```

Arguments

uri   Character variable with a URI describing a cloud bucket

vfs   A TileDB VFS object; default is to use a cached value.

Value

A boolean value indicating if it is an empty bucket

Examples

```
## Not run:
cfg <- tiledb_config()
cfg["vfs.s3.region"] <- "us-west-1"
ctx <- tiledb_ctx(cfg)
vfs <- tiledb_vfs()
tiledb_vfs_is_empty_bucket(vfs, "s3://tiledb-public-us-west-1/test-array-4x4")

## End(Not run)
```

---

tiledb_vfs_is_file   *Test for VFS File*

Description

Test for VFS File

Usage

```
tiledb_vfs_is_file(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

Value

A boolean value indicating if it is a file

---

tiledb_vfs_ls   *Return VFS Directory Listing*

Description

Return VFS Directory Listing

Usage

```
tiledb_vfs_ls(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The content of the directory, non-recursive

```
tiledb_vfs_ls_recursive
```
Recursively list objects from given URI

Description

This functionality is currently limited to S3 URIs.

Usage

```
tiledb_vfs_ls_recursive(
  uri,
  vfs = tiledb_get_vfs(),
  ctx = tiledb_get_context()
)
```

Arguments

uri	Character variable with a URI describing a file path
vfs	(optiona) A TileDB VFS object; default is to use a cached value.
ctx	(optional) A TileDB Ctx object

Value

A data.frame object with two columns for the full path and the object size in bytes

---

tiledb_vfs_move_dir   *Move (or rename) a VFS Directory*

Description

Move (or rename) a VFS Directory

Usage

```
tiledb_vfs_move_dir(olduri, newuri, vfs = tiledb_get_vfs())
```

Arguments

olduri	Character variable with an existing URI describing a directory path
newuri	Character variable with a new desired URI directory path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The newuri value of the moved directory

---

tiledb_vfs_move_file   *Move (or rename) a VFS File*

Description

Move (or rename) a VFS File

Usage

```
tiledb_vfs_move_file(olduri, newuri, vfs = tiledb_get_vfs())
```

Arguments

olduri	Character variable with an existing URI describing a file path
newuri	Character variable with a new desired URI file path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The newuri value of the moved file

---

tiledb_vfs_open   *Open a TileDB VFS Filehandle for reading or writing*

Description

Open a TileDB VFS Filehandle for reading or writing

Usage

```
tiledb_vfs_open(
  binfile,
  mode = c("READ", "WRITE", "APPEND"),
  vfs = tiledb_get_vfs(),
  ctx = tiledb_get_context()
)
```

Arguments

binfile	A character variable describing the (binary) file to be opened
mode	A character variable with value 'READ', 'WRITE' or 'APPEND'
vfs	A TileDB VFS object; default is to use a cached value.
ctx	(optional) A TileDB Ctx object

Value

A TileDB VFS Filehandle object (as an external pointer)

---

tiledb_vfs_read   *Read from a TileDB VFS Filehandle*

Description

This interface currently defaults to reading an integer vector. This is suitable for R objects as a raw
vector used for (de)serialization can be mapped easily to an integer vector. It is also possible to
memcpy to the contiguous memory of an integer vector should other (non-R) data be transferred.

Usage

```
tiledb_vfs_read(fh, offset, nbytes, ctx = tiledb_get_context())
```

Arguments

fh	A TileDB VFS Filehandle external pointer as returned from `tiledb_vfs_open`
offset	A scalar value with the byte offset from the beginning of the file with a of zero.
nbytes	A scalar value with the number of bytes to be read.
ctx	(optional) A TileDB Ctx object

Value

The binary file content is returned as an integer vector.

`tiledb_vfs_remove_bucket`
Remove a VFS Bucket

Description

Remove a VFS Bucket

Usage

```
tiledb_vfs_remove_bucket(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value

`tiledb_vfs_remove_dir` *Remove a VFS Directory*

Description

Remove a VFS Directory

Usage

```
tiledb_vfs_remove_dir(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a diretory path
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value of the removed directory

`tiledb_vfs_remove_file`

Remove a VFS File

Description

Remove a VFS File

Usage

```
tiledb_vfs_remove_file(uri, vfs = tiledb_get_vfs())
```

Arguments

uri   Character variable with a URI describing a file path

vfs   A TileDB VFS object; default is to use a cached value.

Value

The uri value of the removed file

`tiledb_vfs_serialize`   *Serialize an R Object to a VFS-accessible URI*

Description

Serialize an R Object to a VFS-accessible URI

Usage

```
tiledb_vfs_serialize(obj, uri, vfs = tiledb_get_vfs())
```

Arguments

obj   An R object which will be passed to `serialize()`

uri   Character variable with a URI describing a file path to an RDS file

vfs   A TileDB VFS object; default is to use a cached value.

Value

The uri is returned invisibly

tiledb_vfs_sync	*Sync a TileDB VFS Filehandle*

Description

Sync a TileDB VFS Filehandle

Usage

```
tiledb_vfs_sync(fh, ctx = tiledb_get_context())
```

Arguments

fh	A TileDB VFS Filehandle external pointer as returned from tiledb_vfs_open
ctx	(optional) A TileDB Ctx object

Value

The result of the sync operation is returned.

tiledb_vfs_touch	*Touch a VFS URI Resource*

Description

Touch a VFS URI Resource

Usage

```
tiledb_vfs_touch(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a bucket, file or directory
vfs	A TileDB VFS object; default is to use a cached value.

Value

The uri value

---

tiledb_vfs_unserialize

Unserialize an R Object from a VFS-accessible URI

Description

Unserialize an R Object from a VFS-accessible URI

Usage

```
tiledb_vfs_unserialize(uri, vfs = tiledb_get_vfs())
```

Arguments

uri	Character variable with a URI describing a file path to an RDS file
vfs	A TileDB VFS object; default is to use a cached value.

Value

The unserialized object

---

tiledb_vfs_write   *Write to a TileDB VFS Filehandle*

Description

This interface currently defaults to using an integer vector. This is suitable for R objects as the raw vector result from serialization can be mapped easily to an integer vector. It is also possible to memcpy to the contiguous memory of an integer vector should other (non-R) data be transferred.

Usage

```
tiledb_vfs_write(fh, vec, ctx = tiledb_get_context())
```

Arguments

fh	A TileDB VFS Filehandle external pointer as returned from tiledb_vfs_open
vec	An integer vector of content to be written
ctx	(optional) A TileDB Ctx object

Value

The result of the write operation is returned.

---

tile_order,tiledb_array_schema-method

Returns   the   tile   layout   string   associated   with   the
tiledb_array_schema

Description

Returns the tile layout string associated with the tiledb_array_schema

Usage

```
## S4 method for signature 'tiledb_array_schema'
tile_order(object)
```

Arguments

object   A TileDB Schema object

Value

A character string with tile's layout, e.g., "COL_MAJOR".

---

vfs_file   *Create a custom file connection*

Description

Create a custom file connection

Usage

```
vfs_file(description, mode = "", verbosity = 0L)
```

Arguments

description   path to a filename; contrary to rconnection a connection object is not supported.

mode   character string. A description of how to open the connection if it is to be opened upon creation e.g. "rb". Default "" (empty string) means to not open the connection on creation - user must still call open(). Note: If an "open" string is provided, the user must still call close() otherwise the contents of the file aren't completely flushed until the connection is garbage collected.

verbosity   integer value 0, 1, or 2. Default: 0. Set to 0 for no debugging messages, 1 for some high-level messages and verbosity = 2 for all debugging messages.

Details

This vfs_file() connection works like the file() connection in R itself.

This connection works with both ASCII and binary data, e.g. using readLines() and readBin().

Examples

```
## Not run:
tmp <- tempfile()
dat <- as.raw(1:255)
writeBin(dat, vfs_file(tmp))
readBin(vfs_file(tmp),  raw(), 1000)

## End(Not run)
```

[,tiledb_array,ANY-method

Returns a TileDB array, allowing for specific subset ranges.

Description

Heterogenous domains are supported, including timestamps and characters.

Usage

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

Arguments

x	tiledb_array object
i	optional row index expression which can be a list in which case minimum and maximum of each list element determine a range; multiple list elements can be used to supply multiple ranges.
j	optional column index expression which can be a list in which case minimum and maximum of each list element determine a range; multiple list elements can be used to supply multiple ranges.
...	Extra parameters for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default FALSE, currently unused.

Details

This function may still still change; the current implementation should be considered as an initial draft.

Value

The resulting elements in the selected format

[,tiledb_config,ANY-method
Gets a config parameter value

Description

Gets a config parameter value

Usage

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

Arguments

x	tiledb_config object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default FALSE, currently unused.

Value

a config string value if parameter exists, else NA

Examples

```
cfg <- tiledb_config()
cfg["sm.tile_cache_size"]
cfg["does_not_exist"]
```

[,tiledb_filter_list,ANY-method
Returns the filter at given index

Description

Returns the filter at given index

Usage

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

Arguments

x	tiledb_config object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default false.

Value

object tiledb_filter

Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
filter_list[0]
```

[<-,tiledb_array,ANY,ANY,ANY-method

Sets a tiledb array value or value range

Description

This function assigns a right-hand side object, typically a data.frame or something that can be coerced to a data.frame, to a tiledb array.

Usage

```
## S4 replacement method for signature 'tiledb_array,ANY,ANY,ANY'
x[i, j, ...] <- value
```

Arguments

x	sparse or dense TileDB array object
i	parameter row index
j	parameter column index
...	Extra parameter for method signature, currently unused.
value	The value being assigned

Details

For sparse matrices, row and column indices can either be supplied as part of the left-hand side object, or as part of the data.frame provided approrpiate column names.

This function may still still change; the current implementation should be considered as an initial draft.

Value

The modified object

Examples

```
## Not run:
uri <- "quickstart_sparse" ## as created by the other example
arr <- tiledb_array(uri) ## open array
df <- arr[] ## read current content
## First approach: matching data.frame with appriate row and column
newdf <- data.frame(rows = c(1, 2, 2), cols = c(1, 3, 4), a = df$a + 100)
## Second approach: supply indices explicitly
arr[c(1, 2), c(1, 3)] <- c(42, 43) ## two values
arr[2, 4] <- 88 ## or just one

## End(Not run)
```

[<-,tiledb_config,ANY,ANY,ANY-method
Sets a config parameter value

Description

Sets a config parameter value

Usage

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

Arguments

x	tiledb_config object
i	parameter key string
j	parameter key string
value	value to set, will be converted into a stringa

Value

updated tiledb_config object

Examples

```
cfg <- tiledb_config()
cfg["sm.tile_cache_size"]

# set tile cache size to custom value
cfg["sm.tile_cache_size"] <- 100
cfg["sm.tile_cache_size"]
```

Index