

Package: nordstatExtras (via r-universe)

May 19, 2026

Title Shared 'SQLite' Cache Backend for the 'nordstat' Package Family

Version 0.1.0.9004

Description Provides a SQLite-backed cell-level cache that can be used as a drop-in backend by the 'nordstat' family of packages ('rKolada', 'rTrafal', and 'pixieweb'). Designed for multi-user web applications where minimal fetch latency and asynchronous writes are required. Individual statistical values ('`cells') are stored in a gatekeeper schema with a sidecar table for arbitrary metadata dimensions, enabling deduplication across overlapping queries.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports DBI, RSQLite, rlang, tibble, jsonlite, digest

Suggests mirai, testthat (>= 3.0.0), withr

Config/testthat/edition 3

URL <https://github.com/LCHansson/nordstatExtras>

BugReports <https://github.com/LCHansson/nordstatExtras/issues>

Repository <https://lchansson.r-universe.dev>

Date/Publication 2026-05-19 06:16:18 UTC

RemoteUrl <https://github.com/lchansson/nordstatextras>

RemoteRef HEAD

RemoteSha 69aa1ae09eb3b205ddbe72c981f7410ae828b6c2

Contents

nxt_cache_handler	2
nxt_clear	4
nxt_close	5

nxt_expired_searches	5
nxt_flush	6
nxt_gc	6
nxt_is_backend	7
nxt_open	8
nxt_record_search	9
nxt_search	9
nxt_write_async	10

Index	12
--------------	-----------

nxt_cache_handler	<i>Drop-in cache handler for the nordstat family</i>
-------------------	------------------------------------------------------

Description

Produces a closure matching the `cache_handler()` contract used by `rKolada`, `rTrafafa`, and `pixieweb`: a function of `(method, df)` where `method` is one of `"discover"`, `"load"`, or `"store"`. Unlike the per-package `.rds` handlers, this one stores values in a shared SQLite file. Two kinds of cached content are supported:

Usage

```
nxt_cache_handler(
  source,
  entity,
  cache,
  cache_location,
  key_params = list(),
  kind = c("data", "metadata"),
  ttl_days = NXT_DEFAULT_TTL_DAYS,
  reconstruct_extra = list(),
  normalize_extra = list()
)
```

Arguments

<code>source</code>	One of <code>"kolada"</code> , <code>"trafa"</code> , <code>"pixieweb"</code> .
<code>entity</code>	Entity keyword (e.g. <code>"values"</code> , <code>"products"</code> , <code>"tables"</code>).
<code>cache</code>	Logical — whether caching is enabled.
<code>cache_location</code>	Either an <code>nxt_handle</code> from <code>nxt_open()</code> or a path / function-returning-a-path to a <code>.sqlite</code> file.
<code>key_params</code>	Named list of parameters that together with <code>source</code> + <code>entity</code> form a unique cache key.
<code>kind</code>	Either <code>"data"</code> (default, cell-level) or <code>"metadata"</code> (whole-BLOB). Determines the storage/TTL strategy.

ttl_days	Time-to-live in days. Defaults to 30.
reconstruct_extra	Named list of extra arguments forwarded to the reconstruct function — used when trafa/pixieweb need product/alias/etc. at load time. Ignored when kind = "metadata".
normalize_extra	Same for the normalize function, used at store time. Ignored when kind = "metadata" — metadata is serialized whole.

Details

- kind = "data" (default) — statistical data stored at cell granularity with cross-query freshness. Matches the existing rKolada get_values() / rTrafa get_data() / pixieweb get_data() integration.
- kind = "metadata" — whole serialized objects (tibbles or lists) stored as BLOBs on the queries row. Matches the metadata-caching integration for get_kpi(), get_products(), get_tables(), etc. TTL is query-level.

Value

A function with signature `function(method, df = NULL)`.

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)

# Data caching (cell-level)
ch <- nxt_cache_handler(
  source = "kolada", entity = "values", cache = TRUE,
  cache_location = handle,
  key_params = list(kpi = "N03700", period = "2024")
)
ch("discover") # FALSE - nothing cached yet

# Metadata caching (whole-BLOB)
ch_meta <- nxt_cache_handler(
  source = "kolada", entity = "kpi", cache = TRUE,
  cache_location = handle,
  kind = "metadata",
  key_params = list(id = NULL)
)
ch_meta("discover") # FALSE

nxt_close(handle)
unlink(path)
```

nxt_clear	<i>Clear cached queries from the nordstatExtras backend</i>
-----------	-------------------------------------------------------------

Description

Removes queries (and any cells / metadata rows no longer referenced) from the cache. Filters narrow the scope to a single source, entity, or kind.

Usage

```
nxt_clear(handle, source = NULL, entity = NULL, kind = NULL)
```

Arguments

handle	An <code>nxt_handle</code> from <code>nxt_open()</code> .
source	Optional source filter ("kolada", "trafa", "pixieweb").
entity	Optional entity filter.
kind	Optional kind filter ("data" or "metadata"). NULL (default) clears both kinds.

Value

`invisible(NULL)`. Emits a message summarising what was removed.

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)

# Clear all cached data for a specific source
nxt_clear(handle, source = "kolada")

# Clear everything
nxt_clear(handle)

nxt_close(handle)
unlink(path)
```

nxt_close	<i>Close a nordstatExtras cache handle</i>
-----------	--------------------------------------------

Description

Flushes any pending async writes and disconnects the underlying DBI connection.

Usage

```
nxt_close(handle)
```

Arguments

handle A handle returned by [nxt_open\(\)](#).

Value

invisible(NULL)

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)
nxt_close(handle)
unlink(path)
```

nxt_expired_searches	<i>List recorded searches that are due for a refresh</i>
----------------------	----------------------------------------------------------

Description

Returns the (query, source) pairs whose last_run_at is older than max_age days (or their own ttl_days entry, whichever is smaller). A cron job can iterate the result and re-run each search against the live API, passing the fresh entity list back to [nxt_record_search\(\)](#).

Usage

```
nxt_expired_searches(handle, max_age = 30L)
```

Arguments

handle <nxt_handle> from [nxt_open\(\)](#).

max_age Default maximum age in days when a row's own ttl_days is NULL or smaller. Default 30.

Value

A tibble with columns `query`, `source`, `last_run_at` (POSIXct), `t1_days`, `last_hit_count`. Ordered oldest-first.

<code>nxt_flush</code>	<i>Flush pending async writes</i>
------------------------	-----------------------------------

Description

Blocks until all buffered writes for a handle have been committed.

Usage

```
nxt_flush(handle)
```

Arguments

`handle` An `nxt_handle`.

Value

```
invisible(NULL)
```

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)
nxt_flush(handle)
nxt_close(handle)
unlink(path)
```

<code>nxt_gc</code>	<i>Garbage-collect stale cache rows</i>
---------------------	-----------------------------------------

Description

Branches on `queries.kind`:

Usage

```
nxt_gc(handle, max_age_days = NXT_DEFAULT_TTL_DAYS)
```

Arguments

`handle` An `nxt_handle`.
`max_age_days` Maximum age in days. Default 30.

Details

- kind='data': deletes individual cells older than max_age_days, drops junction rows pointing at them, then removes queries that lost every cell. Queries that still have at least one fresh cell are kept.
- kind='metadata': deletes queries whose fetched_at is older than max_age_days (query-level TTL — metadata blobs are opaque and can't benefit from cross-query freshness).

Finally prunes orphan rows from cell_dims and meta_search.

Value

invisible(NULL).

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)

# Remove cells and metadata older than 30 days (default)
nxt_gc(handle)

# Custom TTL: remove anything older than 7 days
nxt_gc(handle, max_age_days = 7)

nxt_close(handle)
unlink(path)
```

nxt_is_backend	<i>Detect whether a cache location should use the nordstatExtras backend</i>
----------------	------------------------------------------------------------------------------

Description

Returns TRUE when cache_location points at a .sqlite/.db file or an existing nxt_handle. Used by rKolada/rTrafal/pixieweb inside their cache_handler() functions to decide whether to delegate.

Usage

```
nxt_is_backend(cache_location)
```

Arguments

cache_location A path, function returning a path, or nxt_handle.

Value

Logical scalar.

Examples

```
nxt_is_backend("my_cache.sqlite") # TRUE
nxt_is_backend("my_cache.db")    # TRUE
nxt_is_backend("/tmp/cache_dir")  # FALSE
nxt_is_backend(tempdir())         # FALSE
```

nxt_open	<i>Open or create a nordstatExtras SQLite cache</i>
----------	-----------------------------------------------------

Description

Opens a SQLite database at path, creating the file and applying the schema if it does not yet exist. Sets WAL mode and pragmas suitable for multi-process read/write access.

Usage

```
nxt_open(path, create = TRUE)
```

Arguments

path	Path to the .sqlite file.
create	Logical. If FALSE, <code>nxt_open()</code> fails when the file does not exist. Default TRUE.

Value

An object of class `nxt_handle` — a thin wrapper around the DBI connection that also carries the path and async-queue identity.

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)
print(handle)
nxt_close(handle)
unlink(path)
```

nxt_record_search	<i>Record a live search so the local index finds it next time</i>
-------------------	-------------------------------------------------------------------

Description

Appends query to the search_keywords column of every entity in entity_ids (deduplicated, whitespace-separated), refreshes the FTS5 index, and UPSERTs the query into recorded_searches with a last_run_at timestamp. The entities must already exist in meta_search — call the source-specific get_* functions with the nordstatExtras cache first.

Usage

```
nxt_record_search(handle, query, source, entity_ids, ttl_days = 30L)
```

Arguments

handle	<nxt_handle> from <code>nxt_open()</code> .
query	Character scalar — the user's search term (kept verbatim in recorded_searches; added as a token to search_keywords).
source	Character scalar — one of "kolada", "pixieweb", "trafa".
entity_ids	Character vector of entity ids to associate with the query (from the respective get_* output's id/name column).
ttl_days	Integer — max age before <code>nxt_expired_searches()</code> returns this search for re-running. Default 30.

Value

Invisibly, the number of meta_search rows updated.

nxt_search	<i>Full-text search across cached metadata</i>
------------	------------------------------------------------

Description

Runs a typeahead-suitable search against the meta_search FTS5 index. Every call to a metadata-caching function (e.g. get_kpi(), get_products(), get_tables()) populates this index as a side effect, so nxt_search() returns matches across all three source packages in a single query.

Usage

```
nxt_search(handle, query, sources = NULL, entity_types = NULL, limit = 50L)
```

Arguments

handle	An <code>nxt_handle</code> from <code>nxt_open()</code> .
query	FTS5 query string. Supports prefix match via <code>term*</code> , phrase match via <code>"exact phrase"</code> , and boolean operators AND/OR.
sources	Optional character vector restricting results to specific sources (e.g. <code>c("kolada", "pixieweb")</code>).
entity_types	Optional character vector restricting results to specific entity types (e.g. <code>c("kpi", "tables")</code>).
limit	Maximum number of results to return. Default 50.

Value

A tibble with columns `source`, `entity_type`, `entity_id`, `title`, `description`, and `rank` (lower = better match). Empty tibble if nothing matches.

Examples

```
path <- tempfile(fileext = ".sqlite")
handle <- nxt_open(path)

# Populate the search index by storing some metadata
ch <- nxt_cache_handler(
  source = "kolada", entity = "kpi", cache = TRUE,
  cache_location = handle, kind = "metadata",
  key_params = list()
)
ch("store", data.frame(
  id = c("N03700", "N01951"),
  title = c("Befolkning totalt", "Bruttoregionprodukt"),
  description = c("Antal invanare", "BRP per capita")
))

# Typeahead search
nxt_search(handle, "bef*")

# Filter by source
nxt_search(handle, "bef*", sources = "kolada")

nxt_close(handle)
unlink(path)
```

Description

Buffers a write for a specific (source, entity, key_params) query and returns immediately. The actual UPSERT into SQLite happens either in a background mirai task or synchronously when the package isn't available.

Usage

```
nxt_write_async(  
  handle,  
  source,  
  entity,  
  key_params,  
  df,  
  normalize_extra = list()  
)
```

Arguments

handle An `nxt_handle` from `nxt_open()`.
source, entity, key_params Identity of the query (same as `nxt_cache_handler()`).
df The tibble to store.
normalize_extra Named list of extra args for the normalizer.

Value

`invisible(handle)`, for pipe-friendliness.

Examples

```
path <- tempfile(fileext = ".sqlite")  
handle <- nxt_open(path)  
  
df <- data.frame(  
  kpi = "N03700", municipality_id = "0180",  
  year = 2024L, value = 103.2  
)  
suppressWarnings(  
  nxt_write_async(handle, "kolada", "values",  
                 key_params = list(kpi = "N03700"), df = df)  
)  
nxt_flush(handle)  
  
nxt_close(handle)  
unlink(path)
```

Index

`nxt_cache_handler`, 2
`nxt_cache_handler()`, 11
`nxt_clear`, 4
`nxt_close`, 5
`nxt_expired_searches`, 5
`nxt_expired_searches()`, 9
`nxt_flush`, 6
`nxt_gc`, 6
`nxt_is_backend`, 7
`nxt_open`, 8
`nxt_open()`, 2, 4, 5, 9–11
`nxt_record_search`, 9
`nxt_record_search()`, 5
`nxt_search`, 9
`nxt_write_async`, 10