

Package: ontoProc2 (via r-universe)

July 15, 2026

Date 2026-07-13

Title Ontology Facilities Based on INCAtools Semantic SQL

Version 0.99.28

Description This package provides ontology facilities based on INCAtools Semantic SQL. Tooling is provided to retrieve and cache SQLite databases representing curated ontologies. ontologyIndex ontology_index instances can be produced. Anthropic's Claude was used in the development of S7 classes and methods and in the production of documentation.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.5.0)

Imports RSQLite, DBI, BiocFileCache, R.utils, dplyr, ontologyIndex, ontologyPlot, graph, Rgraphviz, methods, S7, utils, httr2, xml2, digest

Suggests knitr, testthat, BiocStyle, DT, GO.db, AnnotationDbi, stringr

VignetteBuilder knitr

biocViews Infrastructure, DataRepresentation, Pathways, SingleCell

URL <https://github.com/vjcitn/ontoProc2>

BugReports <https://github.com/vjcitn/ontoProc2/issues>

Config/roxygen2/version 8.0.0

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-07-13 11:58:05 UTC

RemoteUrl <https://github.com/bioc/ontoProc2>

RemoteRef HEAD

RemoteSha 1af885a1dc58511c879ac8426a013c40cf390baa

Contents

bbop_sqlite_db_gz	3
cells_with_pmp	4
cn2tag	4
count_by_prefix	5
count_descendants	5
describe_table	6
disconnect	7
find_by_restriction	7
find_intersection	8
get_ancestors	9
get_ancestors_partonomy	10
get_definition	10
get_descendants	11
get_descendants_partonomy	12
get_direct_edges	12
get_direct_subclasses	13
get_direct_superclasses	14
get_label	14
get_prefix	15
get_present_pmp	16
get_restrictions	16
get_synonyms	17
get_term_info	18
improve_nodes	18
is_connected	19
list_tables	20
make_graphNEL_from_ontology_plot	20
ncit_map	21
onto_plot2	22
PREDICATES	22
print	23
reconnect	24
report	24
retrieve_semsql_conn	25
run_query	26
search_labels	27
semsql_connect	27
semsql_to_oi	28
semsql_url	29
SemsqlConn	30
tag2cn	30
with_connection	31

bbop_sqlite_db_gz	<i>retrieve and cache all filenames of Semantic SQL ontologies available, checking for updated content relative to cache via ETag</i>
-------------------	---

Description

retrieve and cache all filenames of Semantic SQL ontologies available, checking for updated content relative to cache via ETag

Usage

```
bbop_sqlite_db_gz(
  url = "https://s3.amazonaws.com/bbop-sqlite",
  bfc = BiocFileCache::BiocFileCache(),
  rname = "bbop_sqlite_listing",
  timeout_connect = 5,
  timeout_total = 15
)
```

Arguments

url	source address
bfc	an instance of BiocFileCache
rname	a string to use to identify the cached listing
timeout_connect	passed to httr2::req_options connecttimeout parameter
timeout_total	passed to httr2::req_options seconds parameter

Value

a character vector

Note

There is no real etag discipline for the metadata, so full metadata content is extracted on each call, digested, and compared to digest in cache. Cache is updated when there is a discrepancy, except there is a guard against rewriting cache with corrupted or zero-length data at endpoint. Note this code was negotiated at length with claude.ai Sonnet 4.6.

Examples

```
gzs = bbop_sqlite_db_gz()
head(gzs)
length(gzs)
```

cells_with_pmp	<i>produce a table with cells exhibiting given proteins on plasma membrane according to CL</i>
----------------	--

Description

produce a table with cells exhibiting given proteins on plasma membrane according to CL

Usage

```
cells_with_pmp(curies)
```

Arguments

curies a character vector in format "PR:nnnnnnnnnn"

Value

a data.frame with columns cl, celltype, pr, protein

Examples

```
# 2064: microsialin, 1874: KLRB1-like protein
cells_with_pmp(c("PR:000002064", "PR:000001874"))
```

cn2tag	<i>a named vector with mapping from cell type phrase to CURIE for CL.owl of 2025-12-17</i>
--------	--

Description

a named vector with mapping from cell type phrase to CURIE for CL.owl of 2025-12-17

Usage

```
data(cn2tag)
```

Format

names character vector

Examples

```
data("cn2tag", package = "ontoProc2")
cn2tag["Kupffer cell"]
```

count_by_prefix	<i>Count labeled terms grouped by CURIE prefix</i>
-----------------	--

Description

Count labeled terms grouped by CURIE prefix

Usage

```
count_by_prefix(x, ...)
```

Arguments

x	A SemsqConn object.
...	not used.

Value

data.frame with columns prefix and n, ordered by n descending.

Examples

```
goref <- semsql_connect(ontology = "go")
count_by_prefix(goref)
disconnect(goref)
```

count_descendants	<i>Count the number of descendants of a term</i>
-------------------	--

Description

Count the number of descendants of a term

Usage

```
count_descendants(x, term_id, predicate = "rdfs:subClassOf", ...)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
predicate	character(1) predicate to traverse (default "rdfs:subClassOf").
...	not used

Value

integer(1).

Examples

```
goref <- semsql_connect(ontology = "go")
count_descendants(goref, "GO:0006915") # all apoptosis subtypes
disconnect(goref)
```

describe_table	<i>Describe the columns of a table in a SemsqlConn database</i>
----------------	---

Description

Describe the columns of a table in a SemsqlConn database

Usage

```
describe_table(x, table_name, ...)
```

Arguments

x	A SemsqlConn object.
table_name	character(1) name of the table.
...	not used

Value

data.frame with PRAGMA table_info output (columns: cid, name, type, notnull, dflt_value, pk).

Examples

```
goref <- semsql_connect(ontology = "go")
describe_table(goref, "rdfs_label_statement")
disconnect(goref)
```

disconnect	<i>Disconnect a SemsqConn from its database</i>
------------	---

Description

Disconnect a SemsqConn from its database

Usage

```
disconnect(x, quiet = FALSE, ...)
```

Arguments

x	A SemsqConn object.
quiet	logical(1) if TRUE suppresses the disconnection message (default FALSE).
...	not used

Value

The SemsqConn object invisibly.

find_by_restriction	<i>Find terms that have a given OWL someValuesFrom restriction</i>
---------------------	--

Description

Find terms that have a given OWL someValuesFrom restriction

Usage

```
find_by_restriction(  
  x,  
  property,  
  filler,  
  include_filler_descendants = FALSE,  
  ...  
)
```

Arguments

x	A SemsqConn object.
property	character(1) property CURIE (e.g. "BF0:0000050" for part-of).
filler	character(1) filler class CURIE.
include_filler_descendants	logical(1) if TRUE also match subclasses of filler (default FALSE).
...	not used

Value

data.frame with columns id and label.

Examples

```
goref <- semsql_connect(ontology = "go")
# cellular components that are part_of nucleus (GO:0005634)
find_by_restriction(goref, "BF0:0000050", "GO:0005634")
disconnect(goref)
```

find_intersection	<i>Find terms that are descendants of a superclass and have a given restriction</i>
-------------------	---

Description

Find terms that are descendants of a superclass and have a given restriction

Usage

```
find_intersection(x, superclass_id, relation_property, related_to_id, ...)
```

Arguments

x	A SemsqConn object.
superclass_id	character(1) CURIE of the superclass.
relation_property	character(1) property CURIE for the restriction.
related_to_id	character(1) filler CURIE for the restriction.
...	not used

Value

data.frame with columns id and label.

Examples

```
goref <- semsql_connect(ontology = "go")
# CC terms (GO:0005575) that are part_of nucleus (GO:0005634)
find_intersection(goref, "GO:0005575", "BF0:0000050", "GO:0005634")
disconnect(goref)
```

get_ancestors	<i>Get all ancestors of a term via entailed edges</i>
---------------	---

Description

Get all ancestors of a term via entailed edges

Usage

```
get_ancestors(  
  x,  
  term_id,  
  predicates = "rdfs:subClassOf",  
  include_self = FALSE,  
  ...  
)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
predicates	character vector of predicate CURIEs to follow. Defaults to "rdfs:subClassOf". See PREDICATES() for common values.
include_self	logical(1) whether to include the term itself (default FALSE).
...	not used

Value

data.frame with columns id, label, predicate.

Examples

```
goref <- semsql_connect(ontology = "go")  
get_ancestors(goref, "GO:0006915")  
disconnect(goref)
```

get_ancestors_partonomy

Get ancestors traversing both is-a and part-of relationships

Description

Convenience wrapper around `get_ancestors()` that follows both `rdfs:subClassOf` and `BFO:0000050` (part-of) edges.

Usage

```
get_ancestors_partonomy(conn, term_id, include_self = FALSE)
```

Arguments

<code>conn</code>	A <code>SemsqlConn</code> object.
<code>term_id</code>	character(1) CURIE.
<code>include_self</code>	logical(1) whether to include the term itself (default FALSE).

Value

data.frame with columns `id`, `label`, `predicate`.

Examples

```
goref <- semsql_connect(ontology = "go")
get_ancestors_partonomy(goref, "GO:0005739") # mitochondrion
disconnect(goref)
```

get_definition

Get the text definition for a term

Description

Get the text definition for a term

Usage

```
get_definition(x, term_id, ...)
```

Arguments

<code>x</code>	A <code>SemsqlConn</code> object.
<code>term_id</code>	character(1) CURIE.
<code>...</code>	not used

Value

character(1) definition text, or NA_character_ if not found.

Examples

```
goref <- semsql_connect(ontology = "go")
get_definition(goref, "GO:0006915")
disconnect(goref)
```

get_descendants	<i>Get all descendants of a term via entailed edges</i>
-----------------	---

Description

Get all descendants of a term via entailed edges

Usage

```
get_descendants(
  x,
  term_id,
  predicates = "rdfs:subClassOf",
  include_self = FALSE,
  ...
)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
predicates	character vector of predicate CURIEs to follow. Defaults to "rdfs:subClassOf".
include_self	logical(1) whether to include the term itself (default FALSE).
...	not used

Value

data.frame with columns id, label, predicate.

Examples

```
goref <- semsql_connect(ontology = "go")
get_descendants(goref, "GO:0006915")
disconnect(goref)
```

get_descendants_partonomy

Get descendants traversing both is-a and has-part relationships

Description

Convenience wrapper around `get_descendants()` that follows both `rdfs:subClassOf` and `BF0:0000051` (has-part) edges.

Usage

```
get_descendants_partonomy(conn, term_id, include_self = FALSE)
```

Arguments

<code>conn</code>	A <code>SemsqlConn</code> object.
<code>term_id</code>	character(1) CURIE.
<code>include_self</code>	logical(1) whether to include the term itself (default FALSE).

Value

data.frame with columns `id`, `label`, `predicate`.

Examples

```
goref <- semsql_connect(ontology = "go")
get_descendants_partonomy(goref, "GO:0005634") # nucleus sub-components
disconnect(goref)
```

get_direct_edges

Get direct edges in the ontology graph for a term

Description

Get direct edges in the ontology graph for a term

Usage

```
get_direct_edges(
  x,
  term_id,
  direction = c("outgoing", "incoming", "both"),
  ...
)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
direction	character(1) one of "outgoing", "incoming", "both".
...	not used

Value

data.frame with columns subject, subject_label, predicate, predicate_label, object, object_label.

Examples

```
goref <- semsql_connect(ontology = "go")
get_direct_edges(goref, "GO:0006915")
get_direct_edges(goref, "GO:0006915", direction = "both")
disconnect(goref)
```

get_direct_subclasses *Get direct subclasses of a term*

Description

Get direct subclasses of a term

Usage

```
get_direct_subclasses(x, term_id, ...)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
...	not used

Value

data.frame with columns id and label, ordered by label.

Examples

```
goref <- semsql_connect(ontology = "go")
# direct children of "apoptotic process" (GO:0006915)
get_direct_subclasses(goref, "GO:0006915")
disconnect(goref)
```

`get_direct_superclasses`*Get direct superclasses of a term*

Description

Get direct superclasses of a term

Usage

```
get_direct_superclasses(x, term_id, ...)
```

Arguments

<code>x</code>	A SemsqConn object.
<code>term_id</code>	character(1) CURIE.
<code>...</code>	not used

Value

data.frame with columns `id` and `label`, ordered by `label`.

Examples

```
goref <- semsql_connect(ontology = "go")
get_direct_superclasses(goref, "GO:0006915")
disconnect(goref)
```

`get_label`*Get the rdfs:label for a term*

Description

Get the `rdfs:label` for a term

Usage

```
get_label(x, term_id, ...)
```

Arguments

<code>x</code>	A SemsqConn object.
<code>term_id</code>	character(1) CURIE, e.g. "GO:0006915".
<code>...</code>	not used

Value

character(1) label, or NA_character_ if not found.

Examples

```
goref <- semsql_connect(ontology = "go")
get_label(goref, "GO:0006915") # "apoptotic process"
disconnect(goref)
```

get_prefix

Retrieve the ontology prefix from a SemsqlConn

Description

Retrieve the ontology prefix from a SemsqlConn

Usage

```
get_prefix(x, ...)
```

Arguments

x	A SemsqlConn object.
...	not used

Value

character(1) the primary ontology prefix (e.g. "GO").

Examples

```
goref <- semsql_connect(ontology = "go")
get_prefix(goref)
disconnect(goref)
```

get_present_pmp	<i>produce a table with list of proteins from protein ontology identified as present on cell membranes for input cell type CURIEs</i>
-----------------	---

Description

produce a table with list of proteins from protein ontology identified as present on cell membranes for input cell type CURIEs

Usage

```
get_present_pmp(curies)
```

Arguments

curies a character vector in format "CL:nnnnnnn"

Value

a data.frame with columns cl, celltype, pr, protein

Examples

```
get_present_pmp(c("CL:0000091", "CL:0000926"))
```

get_restrictions	<i>Get OWL someValuesFrom restrictions for a term</i>
------------------	---

Description

Get OWL someValuesFrom restrictions for a term

Usage

```
get_restrictions(x, term_id, ...)
```

Arguments

x A SemsqlConn object.
term_id character(1) CURIE.
... not used

Value

data.frame with columns restriction_id, property, property_label, filler, filler_label.

Examples

```
goref <- semsql_connect(ontology = "go")
# mitochondrion (GO:0005739) has part-of restrictions to cell
get_restrictions(goref, "GO:0005739")
disconnect(goref)
```

get_synonyms	<i>Get synonyms for a term</i>
--------------	--------------------------------

Description

Get synonyms for a term

Usage

```
get_synonyms(
  x,
  term_id,
  type = c("all", "exact", "broad", "narrow", "related"),
  ...
)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
type	character(1) synonym scope: one of "all", "exact", "broad", "narrow", "related".
...	not used

Value

data.frame with columns subject, predicate, synonym.

Examples

```
goref <- semsql_connect(ontology = "go")
get_synonyms(goref, "GO:0006915")
get_synonyms(goref, "GO:0006915", type = "exact")
disconnect(goref)
```

get_term_info	<i>Retrieve a summary of information about a term</i>
---------------	---

Description

Retrieve a summary of information about a term

Usage

```
get_term_info(x, term_id, ...)
```

Arguments

x	A SemsqConn object.
term_id	character(1) CURIE.
...	not used

Value

list with elements id, label, definition, synonyms, superclasses, subclasses.

Examples

```
goref <- semsql_connect(ontology = "go")
info <- get_term_info(goref, "GO:0006915")
info$label
info$superclasses
disconnect(goref)
```

improve_nodes	<i>inject linefeeds for node names for graph, with textual annotation from ontology</i>
---------------	---

Description

inject linefeeds for node names for graph, with textual annotation from ontology

Usage

```
improve_nodes(g, ont)
```

Arguments

g	graphNEL instance
ont	instance of ontology from ontologyIndex

Value

graphNEL instance

Examples

```
requireNamespace("Rgraphviz")
requireNamespace("graph")
clcon <- retrieve_semsql_conn("cl")
cl <- semsql_to_oi(clcon)
cl3k <- c(
  "CL:0000492", "CL:0001054", "CL:0000236", "CL:0000625",
  "CL:0000576", "CL:0000623", "CL:0000451", "CL:0000556"
)
p3k <- ontologyPlot::onto_plot(cl, cl3k)
gnel <- make_graphNEL_from_ontology_plot(p3k)
head(graph::nodes(gnel)) # before improving
gnel <- improve_nodes(gnel, cl)
head(graph::nodes(gnel))
```

is_connected	<i>Test whether a SemsqlConn has a valid open connection</i>
--------------	--

Description

Test whether a SemsqlConn has a valid open connection

Usage

```
is_connected(x, ...)
```

Arguments

x	A SemsqlConn object.
...	not used

Value

logical(1).

Examples

```
goref <- semsql_connect(ontology = "go")
is_connected(goref) # TRUE
disconnect(goref)
is_connected(goref) # FALSE
```

list_tables	<i>List tables in a SemsqConn database</i>
-------------	--

Description

List tables in a SemsqConn database

Usage

```
list_tables(x, ...)
```

Arguments

x	A SemsqConn object.
...	not used

Value

character vector of table names.

Examples

```
goref <- semsql_connect(ontology = "go")
list_tables(goref)
disconnect(goref)
```

make_graphNEL_from_ontology_plot	<i>obtain graphNEL from ontology_plot instance of ontologyPlot</i>
----------------------------------	--

Description

obtain graphNEL from ontology_plot instance of ontologyPlot

Usage

```
make_graphNEL_from_ontology_plot(x)
```

Arguments

x	instance of S3 class ontology_plot
---	------------------------------------

Value

instance of S4 graphNEL class

Examples

```

requireNamespace("Rgraphviz")
requireNamespace("graph")
clcon <- retrieve_semsql_conn("cl")
cl <- semsql_to_oi(clcon)
cl3k <- c(
  "CL:0000492", "CL:0001054", "CL:0000236", "CL:0000625",
  "CL:0000576", "CL:0000623", "CL:0000451", "CL:0000556"
)
p3k <- ontologyPlot::onto_plot(cl, cl3k)
gnel <- make_graphNEL_from_ontology_plot(p3k)
gnel <- improve_nodes(gnel, cl)
graph::graph.par(list(nodes = list(shape = "plaintext", cex = .8)))
gnel <- Rgraphviz::layoutGraph(gnel)
Rgraphviz::renderGraph(gnel)

```

ncit_map

a named vector with values rdfs labels in NCI thesaurus, and names the corresponding formal ontology tags

Description

a named vector with values rdfs labels in NCI thesaurus, and names the corresponding formal ontology tags

Usage

```
data(ncit_map)
```

Format

named character vector

Examples

```

data("ncit_map", package = "ontoProc2")
ncit_map["EFO:1000899"]

```

onto_plot2	<i>high-level use of graph/Rgraphviz for rendering ontology relations</i>
------------	---

Description

high-level use of graph/Rgraphviz for rendering ontology relations

Usage

```
onto_plot2(ont, terms2use, cex = 0.8, ...)
```

Arguments

ont	instance of ontology from ontologyIndex
terms2use	character vector
cex	numeric(1) defaults to .8, supplied to Rgraphviz::graph.par
...	passed to onto_plot of ontologyPlot

Value

graphNEL instance (invisibly)

Examples

```
clcon <- retrieve_semsql_conn("cl")
cl <- semsql_to_oi(clcon)
cl3k <- c(
  "CL:0000492", "CL:0001054", "CL:0000236", "CL:0000625",
  "CL:0000576", "CL:0000623", "CL:0000451", "CL:0000556"
)
onto_plot2(cl, cl3k)
```

PREDICATES	<i>Standard predicate CURIEs used in OBO ontologies</i>
------------	---

Description

A named list of commonly used predicate CURIEs in OBO-format ontologies, for use with [get_ancestors\(\)](#), [get_descendants\(\)](#), and related functions.

Usage

```
PREDICATES
```

Format

A named list with elements subclass_of, part_of, has_part, develops_from, located_in, has_characteristic.

Value

a named list

Examples

```
goref <- semsql_connect(ontology = "go")
# apoptotic process (GO:0006915) ancestors via is-a and part-of
anc <- get_ancestors(goref, "GO:0006915",
  predicates = c(PREDICATES$subclass_of, PREDICATES$part_of)
)
head(anc)
disconnect(goref)
```

print

Show method for SemsqlConn. Concise one-line summary displayed when a SemsqlConn object is auto-printed at the R prompt.

Description

Show method for SemsqlConn. Concise one-line summary displayed when a SemsqlConn object is auto-printed at the R prompt.

Usage

```
## S7 method for class <ontoProc2::SemsqlConn>
print(x, ...)
```

Arguments

x	A SemsqlConn object.
...	not used

Value

x, invisibly

reconnect	<i>Reconnect a SemsqConn to its database</i>
-----------	--

Description

Attempts to reconnect a disconnected SemsqConn object to its database. Returns a new SemsqConn; the original cannot be modified in place due to S7 value semantics.

Usage

```
reconnect(x, ...)
```

Arguments

x	A SemsqConn object.
...	not used

Value

A new SemsqConn object with an active connection.

Examples

```
goref <- semsql_connect(ontology = "go")
disconnect(goref)
goref <- reconnect(goref)
disconnect(goref)
```

report	<i>Display a detailed report of a SemsqConn object</i>
--------	--

Description

Displays a verbose, formatted representation of a SemsqConn object including connection status, database statistics (labeled terms, edges, definitions), prefix breakdown, and available key tables. More informative than print(), intended for interactive exploration.

Usage

```
report(object, ...)
```

```
## S7 method for class <ontoProc2::SemsqConn>
report(object, ...)
```

Arguments

object A SemsqlConn object.
 ... additional arguments (currently unused)

Value

The SemsqlConn object invisibly.

Examples

```
goref <- semsql_connect(ontology = "go")
report(goref)
disconnect(goref)
```

retrieve_semsql_conn	<i>return a SQLite connection (read only) to an INCAtools Semantic SQL ontology</i>
----------------------	---

Description

return a SQLite connection (read only) to an INCAtools Semantic SQL ontology

Usage

```
retrieve_semsql_conn(
  ontology = "efo",
  cache = BiocFileCache::BiocFileCache(),
  cacheid = NULL,
  ...
)
```

Arguments

ontology character(1) short string prefixing .db.gz in the INCAtools collection
 cache a BiocFileCache instance, defaulting to BiocFileCache::BiocFileCache()
 cacheid character(1) or NULL; if non-null, the associated SQLite resource will be used from cache
 ... passed to download.file

Value

an RSQLite DBI connection instance

Note

When the cache is searched, the string given as 'ontology' will be prefixed with '^'. This helps avoid confusion between pcl.db and cl.db, for example.

Examples

```
# first time will involve a download and decompression
aionto <- retrieve_semsql_conn("aio")
head(DBI::dbListTables(aionto))
dplyr::tbl(aionto, "class_node") |> head()
```

run_query

Run an arbitrary SQL query against a SemsqlConn database

Description

Run an arbitrary SQL query against a SemsqlConn database

Usage

```
run_query(x, sql, ...)
```

Arguments

x	A SemsqlConn object.
sql	character(1) SQL query string.
...	not used

Value

data.frame with query results.

Examples

```
goref <- semsql_connect(ontology = "go")
run_query(
  goref,
  "SELECT subject, value AS label FROM rdfs_label_statement LIMIT 5"
)
disconnect(goref)
```

search_labels	<i>Search term labels in a SemsqConn database</i>
---------------	---

Description

Search term labels in a SemsqConn database

Usage

```
search_labels(x, pattern, limit = 20L)
```

Arguments

x	A SemsqConn object.
pattern	character(1) substring to match against rdfs:label values (SQL LIKE pattern, case-insensitive on most SQLite builds).
limit	integer(1) maximum number of rows to return (default 20).

Value

data.frame with columns subject and label.

Examples

```
goref <- semsql_connect(ontology = "go")
search_labels(goref, "apoptosis")
disconnect(goref)
```

semsql_connect	<i>Create a SemsqConn connection</i>
----------------	--------------------------------------

Description

Opens a connection to a SemanticSQL SQLite database, either by supplying a direct file path or by referencing a short ontology name that is retrieved and cached via BiocFileCache.

Usage

```
semsql_connect(
  db_path = NULL,
  ontology_prefix = NULL,
  ontology = NULL,
  cache = BiocFileCache::BiocFileCache(),
  validate = TRUE,
  ...
)
```

Arguments

db_path	character(1) or NULL. Path to an existing SQLite database file. Either db_path or ontology must be supplied.
ontology_prefix	character(1) or NULL. Primary CURIE prefix for the ontology (e.g. "CL"). If NULL and ontology is supplied, defaults to toupper(ontology); otherwise auto-detected from the database.
ontology	character(1) or NULL. Short name of an INCAtools ontology (e.g. "cl", "go"). If supplied, retrieve_semsql_conn() is called to locate or download the cached database.
cache	a BiocFileCache instance used when ontology is supplied. Defaults to BiocFileCache::BiocFileCache
validate	logical(1) if TRUE (the default value) the ontology code is checked against available Semantic SQL resources at INCAtools. Set to FALSE if using off line.
...	passed to retrieve_semsql_conn() and ultimately to utils::download.file() .

Value

A [SemsqlConn\(\)](#) object.

Note

The connection has flag SQLITE_RO for read-only access. There will be an attempt to validate the ontology tag that is supplied, against all available Semantic SQL resources available at INCAtools bucket. Function fails if a match cannot be made, which in general requires network access.

Examples

```
# by ontology short name (downloads if not cached)
goref <- semsql_connect(ontology = "go")
goref
disconnect(goref)
```

semsql_to_oi	<i>produce an ontology_index instance from semantic sql sqlite connection</i>
--------------	---

Description

produce an ontology_index instance from semantic sql sqlite connection

Usage

```
semsql_to_oi(con)
```

Arguments

con DBI::dbConnect value for sqlite table

Value

result of ontologyIndex::ontology_index evaluated for the labels and parent-child relations in tables statements and edge of the semantic sql resource

Examples

```
## Not run:
conn <- semsql_connect(ontology = "aio")
oi <- suppressWarnings(semsql_to_oi(conn@con))
names(oi)

## End(Not run)
```

semsql_url	<i>produce INCAtools distribution URL</i>
------------	---

Description

produce INCAtools distribution URL

Usage

```
semsql_url(ontology = "efo")
```

Arguments

ontology short string that is the prefix to .db.gz in the bbop-sqlite collection

Value

a string with URL for INCAtools resource

Examples

```
semsql_url("cl")
```

SemsqlConn	<i>SemsqlConn: S7 connection wrapper for SemanticSQL databases</i>
------------	--

Description

An S7 class that encapsulates a SQLite connection to an ontology database following the SemanticSQL schema. Provides methods for common ontology queries including label lookup, ancestor/descendant traversal, and relationship queries.

Properties:

- con** The DBI connection object (SQLiteConnection)
- db_path** character(1) path to the SQLite database file
- ontology_prefix** character(1) primary ontology prefix, e.g. "CL" for the Cell Ontology

Usage

SemsqlConn(con = NULL, db_path = character(0), ontology_prefix = character(0))

Arguments

- con DBI connection object
- db_path character path to SQLite database file
- ontology_prefix character, e.g., 'CL' for cell ontology

Value

wrapped connection

tag2cn	<i>a named vector with mapping from CURIE to cell type phrase for CL.owl of 2025-12-17</i>
--------	--

Description

a named vector with mapping from CURIE to cell type phrase for CL.owl of 2025-12-17

Usage

data(tag2cn)

Format

names character vector

Examples

```
data("tag2cn", package = "ontoProc2")
tag2cn[c("CL:0000000", "CL:0000006")]
```

with_connection

Execute code with an automatically managed SemsqConn

Description

Opens a connection, evaluates an expression with conn bound to the open SemsqConn, then closes the connection even if an error occurs. Analogous to Python's context manager (with statement).

Usage

```
with_connection(db_path, expr)
```

Arguments

db_path	character(1) path to the SQLite database.
expr	an expression to evaluate; conn is bound to the open SemsqConn within this expression.

Value

the value of expr.

Examples

```
## Not run:
result <- with_connection("cl.db", {
  get_ancestors(conn, "CL:0000540")
})

## End(Not run)
```

Index

* datasets

- cn2tag, [4](#)
- ncit_map, [21](#)
- tag2cn, [30](#)

bbop_sqlite_db_gz, [3](#)

cells_with_pmp, [4](#)

cn2tag, [4](#)

count_by_prefix, [5](#)

count_descendants, [5](#)

describe_table, [6](#)

disconnect, [7](#)

find_by_restriction, [7](#)

find_intersection, [8](#)

get_ancestors, [9](#)

get_ancestors(), [10](#), [22](#)

get_ancestors_partonomy, [10](#)

get_definition, [10](#)

get_descendants, [11](#)

get_descendants(), [12](#), [22](#)

get_descendants_partonomy, [12](#)

get_direct_edges, [12](#)

get_direct_subclasses, [13](#)

get_direct_superclasses, [14](#)

get_label, [14](#)

get_prefix, [15](#)

get_present_pmp, [16](#)

get_restrictions, [16](#)

get_synonyms, [17](#)

get_term_info, [18](#)

improve_nodes, [18](#)

is_connected, [19](#)

list_tables, [20](#)

make_graphNEL_from_ontology_plot, [20](#)

ncit_map, [21](#)

onto_plot2, [22](#)

PREDICATES, [22](#)

PREDICATES(), [9](#)

print, [23](#)

reconnect, [24](#)

report, [24](#)

report, ontoProc2::SemsqlConn-method
(report), [24](#)

retrieve_semsql_conn, [25](#)

retrieve_semsql_conn(), [28](#)

run_query, [26](#)

search_labels, [27](#)

semsql_connect, [27](#)

semsql_to_oi, [28](#)

semsql_url, [29](#)

SemsqlConn, [30](#)

SemsqlConn(), [28](#)

tag2cn, [30](#)

utils::download.file(), [28](#)

with_connection, [31](#)