

Package ‘rladybugdb’

October 16, 2026

Title R Interface to the 'LadybugDB' Embedded Graph Database

Version 0.4.4

Description Provides an R interface to 'LadybugDB', an embedded columnar graph database using the 'Cypher' query language. 'LadybugDB' was formerly known as 'Kuzu'. This package uses native 'Rcpp' bindings to the 'LadybugDB' C library, requiring no 'Python' runtime.

License MIT + file LICENSE

Encoding UTF-8

NeedsCompilation yes

SystemRequirements C++20, CMake (>= 3.15)

Imports DBI (>= 1.2.0), methods, Rcpp (>= 1.0.0), rlang (>= 1.0.0)

LinkingTo Rcpp

Suggests arrow, bit64, DBItest, tibble, igraph, tidygraph, testthat (>= 3.0.0), withr, knitr, rmarkdown

VignetteBuilder knitr

URL <https://github.com/hadimaster65555/rladybugdb>

BugReports <https://github.com/hadimaster65555/rladybugdb/issues>

Config/roxygen2/version 8.1.0

Author Raden Muhammad Hadi [aut, cre]

Maintainer Raden Muhammad Hadi <hadimaster65555@gmail.com>

Repository CRAN

Date/Publication 2026-10-16 12:40:02 UTC

Contents

as.data.frame.lb_result	2
as_arrow_table	3
as_igraph	3
as_tbl_graph	4

as_tibble	5
format.lb_interval	5
Ladybug	6
LadybugConnection-class	6
LadybugDB-DBI-methods	7
ladybugdb_install	8
ladybugdb_is_installed	9
ladybugdb_version	9
LadybugDriver-class	10
LadybugResult-class	10
lb_begin	10
lb_close	11
lb_connection	11
lb_copy_from_arrow	12
lb_copy_from_csv	12
lb_copy_from_df	13
lb_database	13
lb_execute	14
lb_fetch	15
lb_fetch_arrow	16
lb_list_extensions	16
lb_next_result	17
lb_query	17
lb_query_summary	18
lb_result_info	18
lb_set_timeout	19
with_lb_connection	19

Index 21

as.data.frame.lb_result

Convert a LadybugDB result to a data frame

Description

Materialization preserves the current streaming cursor, so it is repeatable and does not change subsequent calls to [lb_fetch\(\)](#).

Usage

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

Arguments

x	An lb_result.
...	Unused.

Value

A `data.frame` containing all query rows, with one column per returned expression. Nested database values are stored in list columns.

as_arrow_table	<i>Convert a LadybugDB result to an Arrow Table</i>
----------------	---

Description

Convert a LadybugDB result to an Arrow Table

Usage

```
as_arrow_table(x, ...)
```

Arguments

x	An <code>lb_result</code> .
...	Unused.

Value

An `arrow::Table` containing all query rows and columns, backed by LadybugDB's Arrow C Data export. The streaming cursor is preserved.

as_igraph	<i>Convert an lb_result to an igraph object</i>
-----------	---

Description

Extracts node and relationship columns from a LadybugDB query result and constructs an `igraph::graph_from_data_frame` object.

Usage

```
as_igraph(x, ...)
```

Arguments

x	An <code>lb_result</code> object.
...	Unused; kept for S3 compatibility.

Details

Node columns must have LadybugDB type `NODE` and relationship columns type `REL/RELATIONSHIP`. When the result contains no structural columns the function aborts with an informative error.

Value

A directed igraph object whose vertices and edges represent the nodes and relationships in the query result. Database properties become vertex and edge attributes; repeated internal identifiers are deduplicated.

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
  with_lb_connection(":memory:", {
    lb_close(lb_execute(
      conn, "CREATE NODE TABLE Person (name STRING, PRIMARY KEY(name))"
    ))
    lb_close(lb_execute(conn, "CREATE REL TABLE Knows (FROM Person TO Person)"))
    lb_close(lb_execute(
      conn, "CREATE (:Person {name:'A'})-[:Knows]->(:Person {name:'B'})"
    ))
    result <- lb_execute(
      conn, "MATCH (a:Person)-[r:Knows]->(b:Person) RETURN a, r, b"
    )
    graph <- as_igraph(result)
    lb_close(result)
    graph
  }, num_threads = 2)
}
```

as_tbl_graph

Convert an lb_result to a tbl_graph (tidygraph)

Description

Convert an lb_result to a tbl_graph (tidygraph)

Usage

```
as_tbl_graph(x, ...)
```

Arguments

x	An lb_result object.
...	Unused.

Value

A tbl_graph object, inheriting from igraph, whose nodes and edges represent the query's nodes and relationships, with their properties available as attributes for tidygraph operations.

as_tibble	<i>Convert a LadybugDB result to a tibble</i>
-----------	---

Description

Convert a LadybugDB result to a tibble

Usage

```
as_tibble(x, ...)
```

Arguments

x	An lb_result.
...	Passed to <code>as.data.frame()</code> .

Value

A tibble (classes `tbl_df`, `tbl`, and `data.frame`) containing all query rows, with one column per returned expression.

format.lb_interval	<i>Format a LadybugDB interval</i>
--------------------	------------------------------------

Description

Format a LadybugDB interval

Usage

```
## S3 method for class 'lb_interval'
format(x, ...)
```

```
## S3 method for class 'lb_interval'
print(x, ...)
```

Arguments

x	An lb_interval value.
...	Unused.

Value

`format()` returns a character string describing the interval's months, days, and microseconds.
`print()` displays that description and returns the supplied `lb_interval` object invisibly.

Examples

```
interval <- structure(
  list(months = 1L, days = 2L, microseconds = 3000000),
  class = "lb_interval"
)
format(interval)
print(interval)
```

Ladybug	Create a LadybugDB DBI driver
---------	-------------------------------

Description

Create a LadybugDB DBI driver

Usage

```
Ladybug()
```

Value

A LadybugDriver S4 object, inheriting from DBIDriver, for use with `DBI::dbConnect()` to open a LadybugDB database.

LadybugConnection-class	LadybugDB DBI connection
-------------------------	--------------------------

Description

LadybugDB DBI connection

Slots

- database Native database object.
- connection Native connection object.
- state Mutable connection state.
- bigint INT64 conversion policy.

Description

LadybugDB implements the standard DBI lifecycle, query, binding, fetch, transaction, quoting, and table-metadata generics. LadybugDB's native schema distinguishes node and relationship tables, so relational helpers outside this documented subset are intentionally left to graph-specific Cypher and the `lb_*` API. SQL `SELECT`, `dbWriteTable()`, and `dbAppendTable()` are not translated. `dbExecute()` returns zero after successful DDL/DML because the LadybugDB 0.20.4 C result summary does not expose a stable affected-row count. Applicable driver behavior is exercised with a focused DBItest subset.

Value

The return value depends on the DBI method:

- `dbConnect()` returns a `LadybugConnection` S4 object, inheriting from `DBIConnection`, that owns an open database and native connection.
- `dbSendQuery()` and `dbSendStatement()` return a `LadybugResult` S4 object, inheriting from `DBIResult`, representing an executed statement or a parameterized statement awaiting `dbBind()`.
- `dbBind()` returns the supplied `LadybugResult` invisibly after executing the statement with the supplied parameters and resetting its fetch cursor.
- `dbFetch()` and `dbGetQuery()` return a `data.frame` with one column per returned expression and one row per query record. `dbFetch()` advances the cursor; `dbGetQuery()` fetches all rows and clears its result.
- `dbExecute()` and `dbGetRowsAffected()` return the numeric scalar `0`. The engine does not expose an affected-row count, so this value does not indicate how many rows were changed.
- `dbGetRowCount()` returns a numeric scalar counting rows fetched since the result was created or most recently bound.
- `dbColumnInfo()` returns a `data.frame` with character columns `name` and `type`, giving each result column's name and LadybugDB type.
- `dbGetInfo()` returns a named list. For a driver, `driver.version` and `client.version` are character strings identifying the package and engine versions. For a connection, `dbname`, `bigint`, and `serverVersion` are character strings describing its path, integer conversion policy, and engine version; `read.only` and `valid` are logical scalars. For a result, `statement` is the Cypher string, `row.count` is the numeric fetched-row count, and `completed` is a logical scalar. An unbound result also has `pending = TRUE`; an executed, valid result also includes the metadata described by `lb_result_info()`.
- `dbIsValid()`, `dbHasCompleted()`, and `dbIsReadOnly()` return a logical scalar indicating whether the object is usable, fetching is complete, or the database was opened read-only, respectively. Drivers are always valid.
- `dbExistsTable()` returns a logical vector indicating whether each supplied table name exists.

- `dbListTables()` and `dbListFields()` return character vectors of table names and field names, respectively; an empty vector means none were found.
- `dbDataType()` returns a character string naming the LadybugDB type for an R vector, or a named character vector of types for data-frame columns.
- `dbQuoteIdentifier()` and `dbQuoteString()` return `DBI::SQL` objects containing quoted identifiers or string literals, with the same length as the input character vector.
- `dbDisconnect()`, `dbClearResult()`, `dbBegin()`, `dbCommit()`, `dbRollback()`, and `dbUnloadDriver()` return the logical scalar `TRUE` after successfully releasing resources or completing transaction control.
- `show()` returns invisible `NULL`, called for the side effect of displaying a summary of the driver, connection, or result.

See Also

[Ladybug\(\)](#), [DBI::DBI-package](#)

ladybugdb_install	<i>Install the LadybugDB C library (deprecated)</i>
-------------------	---

Description

This function is deprecated. The C library is built from bundled source during package installation and no Python installation is required. This helper only issues a deprecation warning; it does not install any software.

Usage

```
ladybugdb_install(...)
```

Arguments

... Ignored.

Value

Invisible `NULL`, called for the side effect of issuing a deprecation warning. No software is installed.

`ladybugdb_is_installed`*Check whether LadybugDB is available*

Description

Returns TRUE if the native LadybugDB C library is loaded and functional.

Usage

```
ladybugdb_is_installed()
```

Value

A logical scalar: TRUE if the native LadybugDB C library responds to a version query, and FALSE if that query fails.

Examples

```
ladybugdb_is_installed()
```

`ladybugdb_version`*Return the LadybugDB C library version string*

Description

Return the LadybugDB C library version string

Usage

```
ladybugdb_version()
```

Value

A character string identifying the bundled LadybugDB engine version, e.g. "0.20.4".

Examples

```
ladybugdb_version()
```

LadybugDriver-class	<i>LadybugDB DBI driver</i>
---------------------	-----------------------------

Description

LadybugDB DBI driver

Slots

state Internal driver state.

LadybugResult-class	<i>LadybugDB DBI result</i>
---------------------	-----------------------------

Description

LadybugDB DBI result

Slots

connection Owning DBI connection.
state Mutable result state.
statement Original Cypher statement.

lb_begin	<i>Transaction control</i>
----------	----------------------------

Description

Transaction control

Usage

lb_begin(conn)

lb_commit(conn)

lb_rollback(conn)

Arguments

conn An lb_connection.

Value

The supplied lb_connection object invisibly, after starting, committing, or rolling back its transaction, respectively.

lb_close	<i>Close a LadybugDB result, connection, or database</i>
----------	--

Description

Closing an object more than once is harmless. A connection cannot be closed while it owns open results, and a database cannot be closed while it owns open connections.

Usage

```
lb_close(x, ...)
```

Arguments

x	An lb_result, lb_connection, or lb_database.
...	Unused.

Value

Invisible NULL, called for the side effect of releasing the object's native resources.

lb_connection	<i>Open a connection to a LadybugDB database</i>
---------------	--

Description

Open a connection to a LadybugDB database

Usage

```
lb_connection(database, num_threads = NULL)
```

Arguments

database	An lb_database object.
num_threads	Positive integer execution thread limit, or NULL.

Value

An lb_connection list containing a native connection pointer and its owning database. Use it to execute queries, and release it with [lb_close\(\)](#) after its results are closed.

lb_copy_from_arrow	<i>Load an Arrow object into a LadybugDB table</i>
--------------------	--

Description

Registers a temporary Arrow-backed table through the Arrow C Data Interface, copies its rows into an existing native node table, and unregisters it. No data-frame or CSV conversion is performed.

Usage

```
lb_copy_from_arrow(conn, value, table)
```

Arguments

conn	An lb_connection.
value	An Arrow Table, RecordBatch, or object accepted by <code>arrow::as_record_batch()</code> .
table	Existing target node table name.

Value

An lb_result containing the COPY statement's metadata after loading the Arrow rows into the target table. Release it with [lb_close\(\)](#).

lb_copy_from_csv	<i>Load a CSV file into a LadybugDB table</i>
------------------	---

Description

Load a CSV file into a LadybugDB table

Usage

```
lb_copy_from_csv(conn, path, table, header = TRUE, delim = ",", ...)
```

Arguments

conn	An lb_connection.
path	Path to a CSV file.
table	Existing target table name.
header	Does the file contain a header?
delim	One-character delimiter.
...	Named COPY options. Values must be scalar logical, numeric, or character values; option names are validated identifiers.

Value

An `lb_result` containing the COPY statement's metadata after loading the CSV rows into the target table. Release it with `lb_close()`.

lb_copy_from_df	<i>Load an R data frame into a LadybugDB table</i>
-----------------	--

Description

Uses native Arrow transport when the optional arrow package is available. Otherwise it uses the explicit CSV loader as a compatibility fallback.

Usage

```
lb_copy_from_df(conn, df, table)
```

Arguments

conn	An <code>lb_connection</code> .
df	A data frame.
table	Existing target table name.

Value

An `lb_result` containing the COPY statement's metadata after loading the data-frame rows into the target table. Release it with `lb_close()`.

lb_database	<i>Open a LadybugDB database</i>
-------------	----------------------------------

Description

Creates or opens a LadybugDB database. LadybugDB v0.20.4 can read the storage formats created by v0.15.x. For incompatible future formats, export with the original engine and import into a fresh database.

Usage

```
lb_database(path = ":memory:", read_only = FALSE, config = list())
```

Arguments

path	File-system path, or ":memory:" for an ephemeral database.
read_only	Open in read-only mode?
config	Named list of LadybugDB system configuration overrides. Valid names are buffer_pool_size, max_num_threads, enable_compression, max_db_size, auto_checkpoint, checkpoint_threshold, throw_on_wal_replay_failure, enable_checksums, enable_multi_writes, and enable_default_hash_index (thread_qos is also accepted on macOS).

Value

An lb_database list containing a native database pointer, its path, read-only flag, and configuration. It represents an open database that should be released with [lb_close\(\)](#) after its connections are closed.

lb_execute	<i>Execute a Cypher query</i>
------------	-------------------------------

Description

Sends a Cypher query string to LadybugDB and returns an lb_result object. For DDL or DML statements (CREATE TABLE, CREATE, MERGE, ...) the result contains metadata; for MATCH/RETURN it contains rows.

Usage

```
lb_execute(conn, query, parameters = NULL)
```

Arguments

conn	An lb_connection object from lb_connection() .
query	A single character string containing a Cypher query.
parameters	A named list of query parameters for parameterised queries. NULL (default) means no parameters.

Value

An lb_result list containing a native result pointer, its owning connection, query text, column names and types, and row count. It represents query rows or statement metadata and must be released with [lb_close\(\)](#).

Examples

```
with_lb_connection(":memory:", {
  lb_close(lb_execute(
    conn, "CREATE NODE TABLE Person (name STRING, PRIMARY KEY(name))"
  ))
  result <- lb_execute(conn, "CREATE (:Person {name: $name})",
    parameters = list(name = "Alice"))
  print(result)
  lb_close(result)
}, num_threads = 2)
```

lb_fetch	<i>Fetch rows from a LadybugDB result</i>
----------	---

Description

Fetch rows from a LadybugDB result

Usage

```
lb_fetch(result, n = 10000)

lb_has_next(result)

lb_reset(result)
```

Arguments

result	An lb_result.
n	Maximum rows to fetch. Inf fetches the remainder.

Value

lb_fetch() returns a data.frame with up to n rows from the current cursor position and one column per returned expression. Repeated calls advance the cursor. lb_has_next() returns a logical scalar that is TRUE when unfetched rows remain. lb_reset() returns the supplied lb_result invisibly after moving its cursor back to the first row.

lb_fetch_arrow	<i>Fetch a chunk as an Arrow Table</i>
----------------	--

Description

Fetch a chunk as an Arrow Table

Usage

```
lb_fetch_arrow(result, n = Inf)
```

Arguments

result	An lb_result.
n	Maximum rows to fetch. Inf fetches all remaining rows.

Value

An arrow::Table containing up to n rows from the current cursor position. The cursor advances by the number of rows returned.

lb_list_extensions	<i>Manage LadybugDB extensions</i>
--------------------	------------------------------------

Description

These are thin wrappers around LadybugDB's extension Cypher commands. Remote installation and updating are unavailable in the CRAN build because those operations require a native networking stack. Compatible extensions installed outside R can still be loaded with lb_load_extension().

Usage

```
lb_list_extensions(conn)

lb_install_extension(conn, name)

lb_load_extension(conn, name)

lb_update_extension(conn, name)
```

Arguments

conn	An lb_connection.
name	Official extension name.

Value

`lb_list_extensions()` returns a `data.frame` describing the official extensions reported by the engine, with one row per extension. `lb_load_extension()` returns an `lb_result` containing the LOAD statement's metadata; release it with `lb_close()`. `lb_install_extension()` and `lb_update_extension()` always raise an error of class `rladybugdb_error_unsupported` and return no value, because remote installation and updates are unavailable in the CRAN build.

<code>lb_next_result</code>	<i>Retrieve the next result from a multi-statement query</i>
-----------------------------	--

Description

Retrieve the next result from a multi-statement query

Usage

```
lb_next_result(result)
```

Arguments

`result` An `lb_result`.

Value

An `lb_result` representing the next statement's rows or metadata, or `NULL` if no further result is available. Close each returned result with `lb_close()`.

<code>lb_query</code>	<i>Execute a Cypher query and return a data frame</i>
-----------------------	---

Description

Convenience wrapper around `lb_execute()` that immediately converts the result to an R `data.frame`. Equivalent to `as.data.frame(lb_execute(conn, query, ...))`.

Usage

```
lb_query(conn, query, parameters = NULL, ...)
```

Arguments

`conn` An `lb_connection` object from `lb_connection()`.
`query` A single character string containing a Cypher query.
`parameters` A named list of query parameters for parameterised queries. `NULL` (default) means no parameters.
`...` Passed to `as.data.frame.lb_result()`.

Value

A data.frame containing all query rows, with one column per returned expression. The underlying native result is closed automatically.

Examples

```
with_lb_connection(":memory:", {
  lb_close(lb_execute(
    conn,
    "CREATE NODE TABLE Person (name STRING, age INT64, PRIMARY KEY(name))"
  ))
  lb_close(lb_execute(conn, "CREATE (:Person {name: 'Alice', age: 30})"))
  lb_query(conn, "MATCH (p:Person) RETURN p.name AS name, p.age AS age")
}, num_threads = 2)
```

<code>lb_query_summary</code>	<i>Return query timing information</i>
-------------------------------	--

Description

Return query timing information

Usage

```
lb_query_summary(result)
```

Arguments

`result` An `lb_result`.

Value

A named list with numeric scalars `compiling_time_ms` and `execution_time_ms`, measuring the query's compilation and execution times in milliseconds.

<code>lb_result_info</code>	<i>Inspect result metadata</i>
-----------------------------	--------------------------------

Description

Inspect result metadata

Usage

```
lb_result_info(result)
```

Arguments

result An lb_result.

Value

A named list with character vectors `column_names` and `column_types`; numeric scalars `num_columns`, `num_tuples` (total rows), and `rows_fetched` (current cursor position); and logical scalars `complete` (no unfetched rows remain) and `has_next_result` (another statement's result is available).

lb_set_timeout	<i>Configure and interrupt queries</i>
----------------	--

Description

Configure and interrupt queries

Usage

```
lb_set_timeout(conn, milliseconds)
```

```
lb_interrupt(conn)
```

Arguments

conn An lb_connection.

milliseconds Non-negative timeout in milliseconds. Zero disables the timeout.

Value

The supplied `lb_connection` object invisibly. `lb_set_timeout()` changes its query timeout; `lb_interrupt()` requests interruption of its running query.

with_lb_connection	<i>Run code with an automatically closed LadybugDB connection</i>
--------------------	---

Description

Run code with an automatically closed LadybugDB connection

Usage

```
with_lb_connection(path = ":memory:", code, ..., num_threads = NULL)
```

Arguments

path	Database path.
code	Code evaluated with conn bound to the open connection.
...	Passed to <code>lb_database()</code> .
num_threads	Passed to <code>lb_connection()</code> .

Value

The value produced by code, with its class and visibility preserved. The connection and database are closed before returning.

Index

`as.data.frame()`, 5
`as.data.frame.lb_result`, 2
`as.data.frame.lb_result()`, 17
`as_arrow_table`, 3
`as_igraph`, 3
`as_tbl_graph`, 4
`as_tibble`, 5

`dbBegin`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbBind`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbClearResult`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbColumnInfo`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbCommit`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbConnect`, LadybugDriver-method
(LadybugDB-DBI-methods), 7
`dbDataType`, LadybugDriver-method
(LadybugDB-DBI-methods), 7
`dbDisconnect`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbExecute`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbExistsTable`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbFetch`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbGetInfo`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbGetInfo`, LadybugDriver-method
(LadybugDB-DBI-methods), 7
`dbGetInfo`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbGetQuery`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbGetRowCount`, LadybugResult-method
(LadybugDB-DBI-methods), 7

`dbGetRowsAffected`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbHasCompleted`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`DBI::dbConnect()`, 6
`DBI::DBI`-package, 8
`dbIsReadOnly`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbIsValid`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbIsValid`, LadybugDriver-method
(LadybugDB-DBI-methods), 7
`dbIsValid`, LadybugResult-method
(LadybugDB-DBI-methods), 7
`dbListFields`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbListTables`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbQuoteIdentifier`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbQuoteString`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbRollback`, LadybugConnection-method
(LadybugDB-DBI-methods), 7
`dbSendQuery`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbSendStatement`, LadybugConnection, character-method
(LadybugDB-DBI-methods), 7
`dbUnloadDriver`, LadybugDriver-method
(LadybugDB-DBI-methods), 7

`format.lb_interval`, 5

`igraph::graph_from_data_frame()`, 3

`Ladybug`, 6
`Ladybug()`, 8
`LadybugConnection`-class, 6
`LadybugDB-DBI`-methods, 7
`ladybugdb_install`, 8

- ladybugdb_is_installed, 9
- ladybugdb_version, 9
- LadybugDriver-class, 10
- LadybugResult-class, 10
- lb_begin, 10
- lb_close, 11
- lb_close(), 11–14, 17
- lb_commit (lb_begin), 10
- lb_connection, 11
- lb_connection(), 14, 17, 20
- lb_copy_from_arrow, 12
- lb_copy_from_csv, 12
- lb_copy_from_df, 13
- lb_database, 13
- lb_database(), 20
- lb_execute, 14
- lb_execute(), 17
- lb_fetch, 15
- lb_fetch(), 2
- lb_fetch_arrow, 16
- lb_has_next (lb_fetch), 15
- lb_install_extension
 - (lb_list_extensions), 16
- lb_interrupt (lb_set_timeout), 19
- lb_list_extensions, 16
- lb_load_extension (lb_list_extensions),
16
- lb_next_result, 17
- lb_query, 17
- lb_query_summary, 18
- lb_reset (lb_fetch), 15
- lb_result_info, 18
- lb_result_info(), 7
- lb_rollback (lb_begin), 10
- lb_set_timeout, 19
- lb_update_extension
 - (lb_list_extensions), 16
- print.lb_interval (format.lb_interval),
5
- show, LadybugConnection-method
 - (LadybugDB-DBI-methods), 7
- show, LadybugDriver-method
 - (LadybugDB-DBI-methods), 7
- show, LadybugResult-method
 - (LadybugDB-DBI-methods), 7
- with_lb_connection, 19