Package: liteq (via r-universe)

June 4, 2024

Title Lightweight Portable Message Queue Using 'SQLite'

Version 1.1.0

Description Temporary and permanent message queues for R. Built on top of 'SQLite' databases. 'SQLite' provides locking, and makes it possible to detect crashed consumers. Crashed jobs can be automatically marked as ``failed'', or put in the queue again, potentially a limited number of times.

License MIT + file LICENSE

URL https://github.com/r-lib/liteq#readme

BugReports https://github.com/r-lib/liteq/issues

Depends R (>= 3.6)

Imports assertthat, DBI, rappdirs, RSQLite

Suggests callr, covr, processx, testthat, withr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://r-lib.r-universe.dev

RemoteUrl https://github.com/r-lib/liteq

RemoteRef HEAD

RemoteSha 961b352e41d68e912c7bde4dde3d3be4f6a2d154

Contents

---

ack *Acknowledge that the work on a message has finished successfully*

Description

Acknowledge that the work on a message has finished successfully

Usage

```
ack(message)
```

Arguments

message	The message object.

See Also

liteq for examples

Other liteq messages: consume(), is_empty(), list_failed_messages(), list_messages(), message_count(), publish(), remove_failed_messages(), requeue_failed_messages(), try_consume()

---

consume *Consume a message from a queue*

Description

Blocks and waits for a message if there isn't one to work on currently.

Usage

```
consume(queue, poll_interval = 500)
```

Arguments

queue	The queue object.
poll_interval	Poll interval in milliseconds. How often to poll the queue for new jobs, if none are immediately available.

Value

A message.

See Also

liteq for examples

Other liteq messages: ack(), is_empty(), list_failed_messages(), list_messages(), message_count(), publish(), remove_failed_messages(), requeue_failed_messages(), try_consume()

---

create_queue *Create a queue in a database*

Description

It also creates the database, if it does not exist.

Usage

```
create_queue(name = NULL, db = default_db(), crash_strategy = "fail")
```

Arguments

name	Name of the queue. If not specified or NULL, a name is generated randomly.
db	Path to the database file.
crash_strategy	What to do with crashed jobs. The default is that they will "fail" (just like a negative acknowledgement). Another possibility is "requeue", in which case they are requeued immediately, potentially even multiple times. Alternatively it can be a number, in which case they are requeued at most the specified number of times.

See Also

liteq for examples

Other liteq queues: delete_queue(), ensure_queue(), list_queues()

default_db *The name of the default database*

Description

If the queue database is not specified explicitly, then `liteq` uses this file. Its location is determined via the `rappdirs` package, see `rappdirs::user_data_dir()`.

Usage

```
default_db()
```

Value

A characater scalar, the name of the default database.

delete_queue *Delete a queue*

Description

Delete a queue

Usage

```
delete_queue(queue, force = FALSE)
```

Arguments

`queue`	The queue to delete.
`force`	Whether to delete the queue even if it contains messages.

See Also

liteq for examples

Other liteq queues: `create_queue()`, `ensure_queue()`, `list_queues()`

ensure_queue *Make sure that a queue exists*

Description

If it does not exist, then the queue will be created.

Usage

```
ensure_queue(name, db = default_db(), crash_strategy = "fail")
```

Arguments

name	Name of the queue. If not specified or NULL, a name is generated randomly.
db	Path to the database file.
crash_strategy	What to do with crashed jobs. The default is that they will "fail" (just like a negative acknowledgement). Another possibility is "requeue", in which case they are requeued immediately, potentially even multiple times. Alternatively it can be a number, in which case they are requeued at most the specified number of times.

Value

The queue object.

See Also

liteq for examples

Other liteq queues: create_queue(), delete_queue(), list_queues()

is_empty *Check if a queue is empty*

Description

Check if a queue is empty

Usage

```
is_empty(queue)
```

Arguments

queue	The queue object.

Value

Logical, whether the queue is empty.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `list_failed_messages()`, `list_messages()`, `message_count()`, `publish()`, `remove_failed_messages()`, `requeue_failed_messages()`, `try_consume()`

`list_failed_messages` *List failed messages in a queue*

Description

List failed messages in a queue

Usage

```
list_failed_messages(queue)
```

Arguments

queue	The queue object.

Value

Data frame with columns: `id`, `title`, `status`.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_messages()`, `message_count()`, `publish()`, `remove_failed_messages()`, `requeue_failed_messages()`, `try_consume()`

list_messages *List all messages in a queue*

Description

List all messages in a queue

Usage

```
list_messages(queue)
```

Arguments

queue	The queue object.

Value

Data frame with columns: `id`, `title`, `status`.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_failed_messages()`, `message_count()`, `publish()`, `remove_failed_messages()`, `requeue_failed_messages()`, `try_consume()`

list_queues *List all queues in a database*

Description

List all queues in a database

Usage

```
list_queues(db = default_db())
```

Arguments

db	The queue database to query.

Value

A list of `liteq_queue` objects.

See Also

liteq for examples

Other liteq queues: `create_queue()`, `delete_queue()`, `ensure_queue()`

`liteq` *Lightweight Portable Message Queue Using 'SQLite'*

Description

Message queues for R. Built on top of 'SQLite' databases.

Concurrency

liteq works with multiple producer and/or consumer processes accessing the same queue, via the locking mechanism of 'SQLite'. If a queue is locked by 'SQLite', the process that tries to access it, must wait until it is unlocked. The maximum amount of waiting time is by default 10 seconds, and it can be changed via the `R_LITEQ_BUSY_TIMEOUT` environment variable, in milliseconds. If you have many concurrent processes using the same liteq database, and see `database locked` errors, then you can try to increase the timeout value.

Examples

```
# We don't run this, because it writes to the cache directory
db <- tempfile()
q <- ensure_queue("jobs", db = db)
q
list_queues(db)

# Publish two messages
publish(q, title = "First message", message = "Hello world!")
publish(q, title = "Second message", message = "Hello again!")
is_empty(q)
message_count(q)
list_messages(q)

# Consume one
msg <- try_consume(q)
msg

ack(msg)
list_messages(q)
msg2 <- try_consume(q)
nack(msg2)
list_messages(q)

# No more messages
is_empty(q)
try_consume(q)
```

Examples

```
## See the manual page
```

`message_count` *Get the number of messages in a queue.*

Description

Get the number of messages in a queue.

Usage

```
message_count(queue)
```

Arguments

queue	The queue object.

Value

Number of messages in the queue.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_failed_messages()`, `list_messages()`, `publish()`, `remove_failed_messages()`, `requeue_failed_messages()`, `try_consume()`

`nack` *Report that the work on a message has failed*

Description

Report that the work on a message has failed

Usage

```
nack(message)
```

Arguments

message	The message object.

See Also

liteq for examples

`publish` *Publish messages in a queue*

Description

Publish messages in a queue

Usage

```
publish(queue, title = "", message = "")
```

Arguments

`queue`	The queue object.
`title`	The title of the messages. It can be the empty string.
`message`	The body of the messages. It can be the empty string. Must be the same length as `title`.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_failed_messages()`, `list_messages()`, `message_count()`, `remove_failed_messages()`, `requeue_failed_messages()`, `try_consume()`

`remove_failed_messages` *Remove failed messages from the queue*

Description

Remove failed messages from the queue

Usage

```
remove_failed_messages(queue, id = NULL)
```

Arguments

`queue`	The queue object.
`id`	Ids of the messages to requeue. If it is `NULL`, then all failed messages will be removed.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_failed_messages()`, `list_messages()`, `message_count()`, `publish()`, `requeue_failed_messages()`, `try_consume()`

`requeue_failed_messages` *Requeue failed messages*

Description

Requeue failed messages

Usage

```
requeue_failed_messages(queue, id = NULL)
```

Arguments

`queue`	The queue object.
`id`	Ids of the messages to requeue. If it is `NULL`, then all failed messages will be requeued.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_failed_messages()`, `list_messages()`, `message_count()`, `publish()`, `remove_failed_messages()`, `try_consume()`

`try_consume` *Consume a message if there is one available*

Description

Consume a message if there is one available

Usage

```
try_consume(queue)
```

Arguments

`queue`	The queue object.

Value

A message, or `NULL` if there is not message to work on.

See Also

liteq for examples

Other liteq messages: `ack()`, `consume()`, `is_empty()`, `list_failed_messages()`, `list_messages()`, `message_count()`, `publish()`, `remove_failed_messages()`, `requeue_failed_messages()`

Index