Package 'dbx'

July 22, 2025

Type Package

Title A Fast, Easy-to-Use Database Interface

Version 0.4.0

Date 2025-03-17

Description Provides select, insert, update, upsert, and delete database operations. Supports 'PostgreSQL', 'MySQL', 'SQLite', and more, and plays nicely with the 'DBI' package.

URL https://github.com/ankane/dbx

BugReports https://github.com/ankane/dbx/issues

License MIT + file LICENSE

RoxygenNote 7.3.1

Encoding UTF-8

Imports DBI (>= 1.0.0)

Suggests testthat (>= 1.0.2), urltools (>= 1.7.0), RSQLite (>= 2.1.2), RMariaDB, RMySQL (>= 0.10.20), RPostgres, RPostgreSQL, hms, jsonlite, blob, odbc

NeedsCompilation no

Author Andrew Kane [aut, cre]

Maintainer Andrew Kane <andrew@chartkick.com>

Repository CRAN

Date/Publication 2025-03-18 11:00:02 UTC

Contents

`dbxConnect` *Create a database connection*

Description

Create a database connection

Usage

```
dbxConnect(
  url = NULL,
  adapter = NULL,
  storage_tz = NULL,
  variables = list(),
  ...
)
```

Arguments

`url`	A database URL
`adapter`	The database adapter to use
`storage_tz`	The time zone timestamps are stored in
`variables`	Session variables
`...`	Arguments to pass to dbConnect

Examples

```
# SQLite
db <- dbxConnect(adapter="sqlite", dbname=":memory:")

## Not run:

# Postgres
db <- dbxConnect(adapter="postgres", dbname="mydb")

# MySQL
db <- dbxConnect(adapter="mysql", dbname="mydb")

# Others
db <- dbxConnect(adapter=odbc(), database="mydb")

## End(Not run)
```

dbxDelete *Delete records*

Description

Delete records

Usage

```
dbxDelete(conn, table, where = NULL, batch_size = NULL)
```

Arguments

conn	A DBIConnection object
table	The table name to delete records from
where	A data frame of records to delete
batch_size	The number of records to delete in a single statement (defaults to all)

Examples

```
db <- dbxConnect(adapter="sqlite", dbname=":memory:")
table <- "forecasts"
DBI::dbCreateTable(db, table, data.frame(id=1:3, temperature=20:22))

# Delete specific records
bad_records <- data.frame(id=c(1, 2))
dbxDelete(db, table, where=bad_records)

# Delete all records
dbxDelete(db, table)
```

dbxDisconnect *Close a database connection*

Description

Close a database connection

Usage

```
dbxDisconnect(conn)
```

Arguments

conn	A DBIConnection object

Examples

```
db <- dbxConnect(adapter="sqlite", dbname=":memory:")

dbxDisconnect(db)
```

`dbxExecute` *Execute a statement*

Description

Execute a statement

Usage

```
dbxExecute(conn, statement, params = NULL)
```

Arguments

`conn`	A DBIConnection object
`statement`	The SQL statement to use
`params`	Parameters to bind

Examples

```
db <- dbxConnect(adapter="sqlite", dbname=":memory:")
DBI::dbCreateTable(db, "forecasts", data.frame(id=1:3, temperature=20:22))

dbxExecute(db, "UPDATE forecasts SET temperature = 20")

dbxExecute(db, "UPDATE forecasts SET temperature = ?", params=list(20))

dbxExecute(db, "UPDATE forecasts SET temperature = ? WHERE id IN (?)", params=list(20, 1:3))
```

`dbxInsert` *Insert records*

Description

Insert records

Usage

```
dbxInsert(conn, table, records, batch_size = NULL, returning = NULL)
```

Arguments

conn	A DBIConnection object
table	The table name to insert
records	A data frame of records to insert
batch_size	The number of records to insert in a single statement (defaults to all)
returning	Columns to return

Examples

```
db <- dbxConnect(adapter="sqlite", dbname=":memory:")
table <- "forecasts"
DBI::dbCreateTable(db, table, data.frame(id=1:3, temperature=20:22))

records <- data.frame(temperature=c(32, 25))
dbxInsert(db, table, records)
```

---

dbxSelect *Select records*

Description

Select records

Usage

```
dbxSelect(conn, statement, params = NULL)
```

Arguments

conn	A DBIConnection object
statement	The SQL statement to use
params	Parameters to bind

Examples

```
db <- dbxConnect(adapter="sqlite", dbname=":memory:")
DBI::dbCreateTable(db, "forecasts", data.frame(id=1:3, temperature=20:22))

dbxSelect(db, "SELECT * FROM forecasts")

dbxSelect(db, "SELECT * FROM forecasts WHERE id = ?", params=list(1))

dbxSelect(db, "SELECT * FROM forecasts WHERE id IN (?)", params=list(1:3))
```

dbxUpdate *Update records*

Description

Update records

Usage

```
dbxUpdate(
  conn,
  table,
  records,
  where_cols,
  batch_size = NULL,
  transaction = TRUE
)
```

Arguments

conn	A DBIConnection object
table	The table name to update
records	A data frame of records to insert
where_cols	The columns to use for WHERE clause
batch_size	The number of records to update in a single transaction (defaults to all)
transaction	Wrap the update in a transaction (defaults to true)

Examples

```
db <- dbxConnect(adapter="sqlite", dbname=":memory:")
table <- "forecasts"
DBI::dbCreateTable(db, table, data.frame(id=1:3, temperature=20:22))

records <- data.frame(id=c(1, 2), temperature=c(16, 13))
dbxUpdate(db, table, records, where_cols=c("id"))
```

dbxUpsert *Upsert records*

Description

Upsert records

Usage

```
dbxUpsert(
  conn,
  table,
  records,
  where_cols,
  batch_size = NULL,
  returning = NULL,
  skip_existing = FALSE
)
```

Arguments

conn	A DBIConnection object
table	The table name to upsert
records	A data frame of records to upsert
where_cols	The columns to use for WHERE clause
batch_size	The number of records to upsert in a single statement (defaults to all)
returning	Columns to return
skip_existing	Skip existing rows

Examples

```
## Not run:

db <- dbxConnect(adapter="postgres", dbname="dbx")
table <- "forecasts"
DBI::dbCreateTable(db, table, data.frame(id=1:3, temperature=20:22))

records <- data.frame(id=c(3, 4), temperature=c(20, 25))
dbxUpsert(db, table, records, where_cols=c("id"))

## End(Not run)
```

Index