

Diesel ORM Cheat Sheet

ORM Rust type-safe pour PostgreSQL / MySQL / SQLite

Installation

CLI Diesel :

```
1 cargo install diesel_cli
```

PostgreSQL :

```
1 cargo install diesel_cli \
2 --no-default-features \
3 --features postgres
```

Dépendances :

```
1 cargo add diesel
2 cargo add dotenvy
3 cargo add r2d2
```

Initialisation

Créer projet DB :

```
1 diesel setup
```

Fichier .env :

```
1 DATABASE_URL=
2 postgres://user:pass@
3 localhost/db
```

Commandes CLI

Commandes principales :

```
1 diesel setup
2 diesel database setup
3 diesel database reset
```

Schema :

```
1 diesel print-schema \
2 > src/schema.rs
```

Migrations

Créer migration :

```
1 diesel migration generate
2 create_users
```

Appliquer :

```
1 diesel migration run
```

Rollback :

```
1 diesel migration revert
```

Redo :

```
1 diesel migration redo
```

Lister :

```
1 diesel migration list
```

En attente :

```
1 diesel migration pending
```

Structure Migration

migrations/

```
000_create_users/
  up.sql
  down.sql
```

up.sql :

```
1 CREATE TABLE users(
2 id SERIAL PRIMARY KEY,
3 name VARCHAR NOT NULL
4 );
```

down.sql :

```
1 DROP TABLE users;
```

Schema

```
1 table! {
2 users (id) {
3
4 id -> Int4,
5 name -> Varchar,
6
7 }
8 }
```

Import :

```
1 use crate::schema::*;
```

Connexion

```
1 use diesel::prelude::*;
2 use diesel::pg::PgConnection;
3
4 fn connect()
5 -> PgConnection {
6
7 let url =
8 env::var("DATABASE_URL")
9 .expect("db");
10
11 PgConnection::establish(&url)
12 .expect("error")
13
14 }
```

Connection Pool

```
1 use diesel::r2d2::{
2 ConnectionManager,
3 Pool
4 };
5
6 type DbPool =
7 Pool<ConnectionManager<
8 PgConnection>>;
```

Models

Queryable :

```
1 #[derive(Queryable)]
2 struct User {
3
4 id: i32,
5 name: String
6
7 }
```

Insertable :

```
1 #[derive(Insertable)]
2 #[diesel(table_name=users)]
3 struct NewUser<'a>{
4
5 name: &'a str
6
7 }
```

Macros Diesel

Macros importantes :

```
1 Queryable
2 Insertable
3 Identifiable
4 Associations
5 AsChangeset
6 Selectable
```

Select

Tous :

```
1 users.load::<User>(&conn)?;
```

Premier :

```
1 users.first(&conn)
```

Find :

```
1 users.find(1)
```

Insert

```
1 insert_into(users)
2 .values(&new_user)
3 .execute(&conn)?;
```

Batch :

```
1 insert_into(users)
2 .values(&vec_users)
3 .execute(&conn)?;
```

Update

```
1 update(users.find(1))
2 .set(name.eq("Bob"))
3 .execute(&conn)?;
```

Delete

```
1 delete(users.find(1))
2 .execute(&conn)?;
```



Filtres

```
1 users
2 .filter(name.eq("Alice"))
3 .load::(&conn)?;
```

Opérateurs :

```
1 eq
2 ne
3 gt
4 lt
5 ge
6 le
7 like
```

Order / Limit

```
1 users
2 .order(id.desc())
3 .limit(10)
4 .offset(20)
```

Joins

```
1 users
2 .inner_join(posts)
3 .load::<(User, Post)>
4 (&conn)?;
```

Left join :

```
1 left_join(posts)
```

Relations

```
1 #[derive(Asociations)]
2 #[diesel(belongs_to(User))]
3 struct Post {
4
5 id: i32,
6 user_id: i32,
7 title: String
8
9 }
```

Transaction

```
1 conn.transaction(|| {
2
3 insert_into(users)
4 .values(&new_user)
5 .execute(&conn)?;
6
7 Ok(())
8
9 });
```

SQL Brut

```
1 use diesel::sql_query;
2
3 sql_query(
4 "SELECT * FROM users"
5 )
6 .load::(&conn)?;
```

