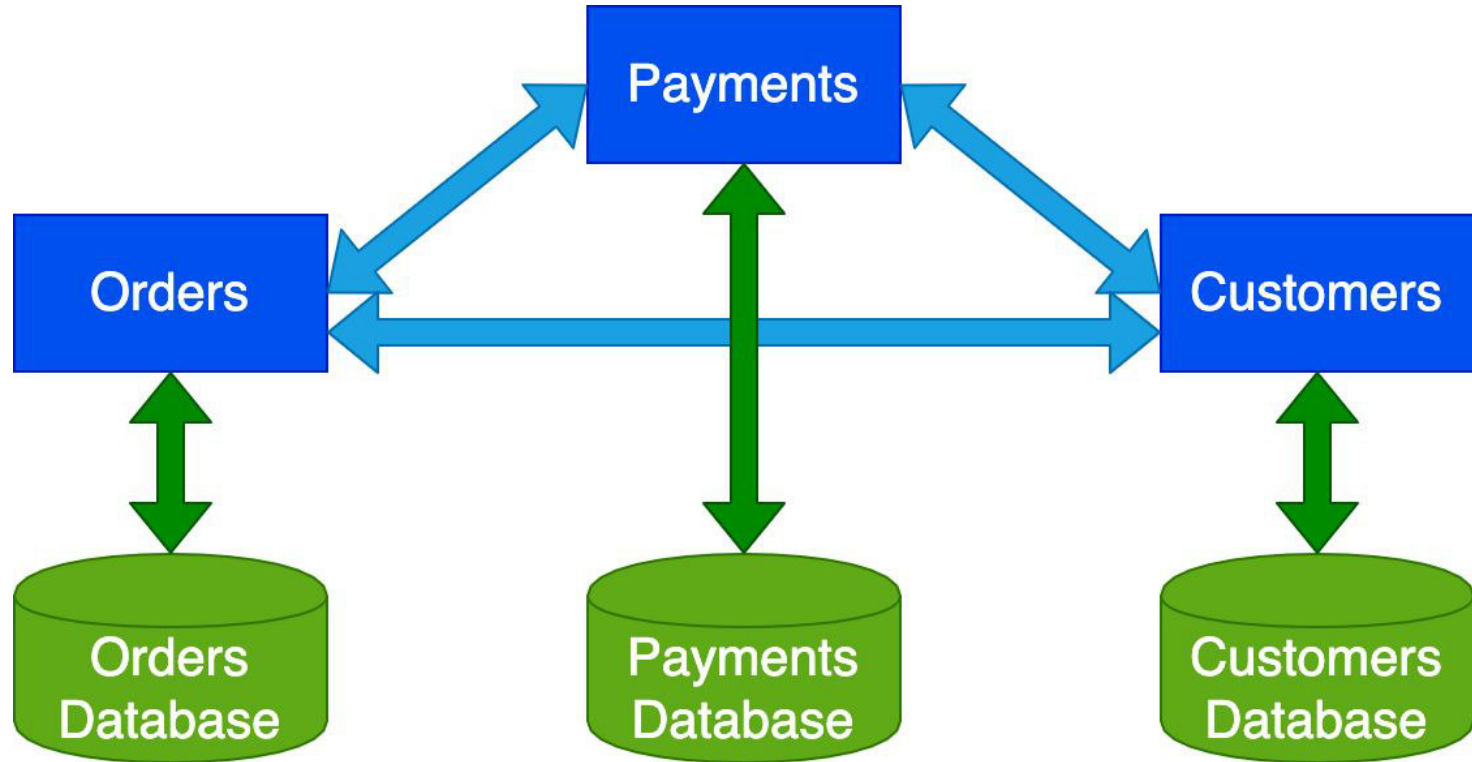


# Работа с миграциями в Go

---

# Database per service



# На что смотреть

## 1. Живость проекта

# На что смотреть

1. Живость проекта
2. Поддерживаемые БД

# На что смотреть

1. Живость проекта
2. Поддерживаемые БД
3. Помощь в избежании ошибок

# На что смотреть

1. Живость проекта
2. Поддерживаемые БД
3. Помощь в избежании ошибок
4. Автоматизация

# golang-migrate/migrate

Watch ▾

51

★ Star

3.6k

🔗 Fork

360

- PostgreSQL, MySQL, SQLite, MS SQL, MongoDB, Cassandra ...
- миграции определяются с помощью sql или go
- используется как cli инструмент или go-библиотека
- применяет и отменяет изменения атомарно
- доступные команды: create, goto V, up [N], down [N], drop, version, force V

# golang-migrate/migrate

## Идеальный кейс

```
-- 000001_init_db.up.sql
```

```
create table product(...);
```

```
insert into product values (1, 'test', 1, 'test', 1);
```

```
insert into product values (2, 'test', 2, 'test', 2);
```

```
-- 000001_init_db.down.sql
```

```
drop table product;
```



# golang-migrate/migrate

Идеальный кейс

```
$ migrate up
```

```
> 1/u init_db (36.645031ms)
```

# golang-migrate/migrate

## Идеальный кейс

```
$ migrate up
```

```
> 1/u init_db (36.645031ms)
```

```
simple=# select * from schema_migrations;  
version | dirty
```

```
-----+-----
```

```
1 | f
```

# golang-migrate/migrate

## Неидеальный кейс

```
-- 000001_init_db.up.sql
```

```
create table product(...);
```

```
insert into product values (1, 'test', 1, 'test', 1);
```

```
insert into product values (1, 'test', 2, 'test', 2);
```

```
create table attribute(...);
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate up
```

```
> error: migration failed: duplicate key value violates  
unique constraint "product_pkey", Key (id)=(1) already  
exists. in line 0: -- 000001_init_db.up.sql
```

# golang-migrate/migrate

## Неидеальный кейс

```
simple=# select * from schema_migrations;
```

```
version | dirty
```

```
-----+-----
```

```
1 | t
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate up
```

```
> error: Dirty database version 1. Fix and force  
version.
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate up
```

```
> error: Dirty database version 1. Fix and force  
version.
```

```
$ migrate force 0
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate up
```

```
> error: Dirty database version 1. Fix and force  
version.
```

```
$ migrate force 0
```

```
simple=# select * from schema_migrations;
```

```
  version | dirty
```

```
-----+-----
```

```
    0 | f
```



# golang-migrate/migrate

Неидеальный кейс

```
$ migrate up
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate up
```

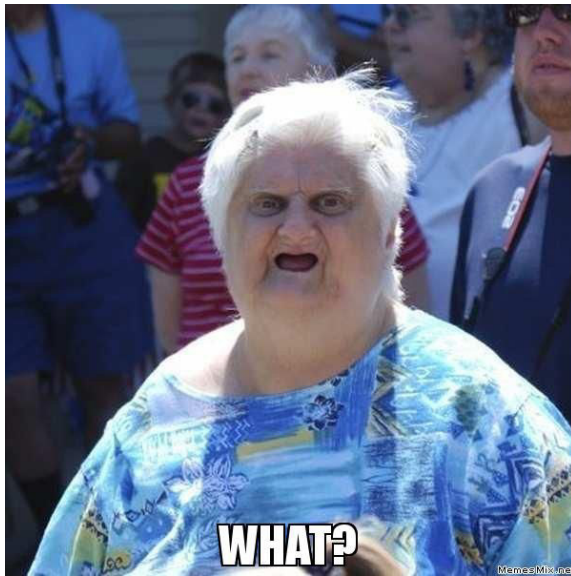
```
> error: no migration found for version 0error: file  
does not exist
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate up
```

```
> error: no migration found for version 0error: file  
does not exist
```



# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate drop
```

```
$ migrate up
```

```
> 1/u init_db (37.603507ms)
```

# golang-migrate/migrate

## Неидеальный кейс

```
$ migrate drop  
$ migrate up  
> 1/u init_db (37.603507ms)
```

or

```
$ migrate force 1  
$ migrate down 1  
$ migrate up  
> 1/u init_db (37.603507ms)
```

# golang-migrate/migrate

## Неидеальный кейс [2]

```
-- 000001_init_db.down.sql
```

```
drop table attribute;
```

```
drop table products;
```

# golang-migrate/migrate

## Неидеальный кейс [2]

```
$ migrate down
```

```
> Applying all down migrations
```

```
error: migration failed: table "products" does not exist  
in line 0: -- 000001_init_db.down.sql
```

```
...
```

# golang-migrate/migrate

## Неидеальный кейс [2]

Schema	Name	Type	Owner
public	attribute	table	test
public	product	table	test
public	schema_migrations	table	test



# golang-migrate/migrate

## Неидеальный кейс [2]

```
simple=# select * from schema_migrations;  
version | dirty
```

```
-----+-----
```

```
(0 rows)
```

# golang-migrate/migrate

Неидеальный кейс [3]

000001\_init\_db.up.sql

**000002\_add\_something.up.sql**

# golang-migrate/migrate

Неидеальный кейс [3]

000001\_init\_db.up.sql

**000002\_add\_something.up.sql**

000002\_add\_test\_table.up.sql

# golang-migrate/migrate

## Неидеальный кейс [3]

```
$ migrate up
```

```
> error: duplicate migration file:  
000002_add_something.up.sql
```

# golang-migrate/migrate

## Неидеальный кейс [3]

```
$ migrate up
```

```
> error: duplicate migration file:  
000002_add_something.up.sql
```

```
simple=# select * from schema_migrations;  
version | dirty  
-----+-----  
2 | f
```

# golang-migrate/migrate

Неидеальный кейс [3]

20200127205541\_init\_db.up.sql

**20200127205542\_add\_something.up.sql**

20200127205545\_add\_test\_table.up.sql

# golang-migrate/migrate

Неидеальный кейс [3]

```
$ migrate up
```

```
> no change
```

# golang-migrate/migrate

## Неидеальный кейс [3]

```
$ migrate up  
> no change
```

```
simple=# select * from schema_migrations;  
      version      | dirty  
-----+-----  
 20200127205545   | f
```



# golang-migrate/migrate

- большое количество поддерживаемых БД
- возможность читать файлы миграции из различных источников
- можно использовать как cli инструмент или библиотеку
- go и sql миграции
- изменения применяются и отменяются атомарно
- сложно автоматизировать
- нужно быть осторожным с откатами и с временной нумерацией версий

# rubenv/sql-migrate

Watch ▾

30

★ Star

1.6k

🔗 Fork

146

- SQLite, PostgreSQL, MySQL, MS SQL, Oracle
- используется как cli инструмент или go-библиотека
- миграции определяются с помощью sql
- конфигурация хранится в отдельном файле
- изменения применяются и отменяются атомарно
- доступные команды: down, new, redo, status, up

# rubenv/sql-migrate

#dbconfig.yml

development:

dialect: sqlite3

datasource: test.db

dir: migration/migrations

production:

dialect: postgres

datasource: postgres://test:111@localhost:5432/simple

dir: /app/migration\_db

table: migrations

# rubenv/sql-migrate

## Ошибка в запросе

```
-- 20200127205541-init_db.sql

-- +migrate Up
create table product(...);
insert into product values (1, 'test', 1, 'test', 1);
insert into product values (1, 'test', 2, 'test', 2);
create table attribute(...);
-- +migrate Down
drop table product;
drop table attribute;
```

# rubenv/sql-migrate

Ошибка в запросе

```
$ sql-migrate up
```

```
> Migration failed: pq: duplicate key value violates  
unique constraint "product_pkey" handling  
20200127205541-init_db.sql
```

# rubenv/sql-migrate

Ошибка в запросе

```
simple=# select * from migrations;
```

```
id | applied_at
```

```
----+-----
```

```
(0 rows)
```

# rubenv/sql-migrate

Исправили ошибку

```
$ sql-migrate up
```

```
> Applied 1 migration
```

# rubenv/sql-migrate

Исправили ошибку

```
$ sql-migrate up  
> Applied 1 migration
```

```
simple=# select * from migrations;
```

id	applied_at
20200127205541-init_db.sql	2020-02-02 15:59:13.596424+00

(1 row)



# rubenv/sql-migrate

Ошибка в запросе для отката

```
-- 20200127205541-init_db.sql
```

```
...
```

```
-- +migrate Down
```

```
drop table product;
```

```
drop table attributes;
```

# rubenv/sql-migrate

Ошибка в запросе для отката

```
$ sql-migrate down
```

```
> Migration failed: pq: table "attributes" does not exist  
handling 20200127205541-init_db.sql
```

# rubenv/sql-migrate

## Ошибка в запросе для отката

```
$ sql-migrate down
```

```
> Migration failed: pq: table "attributes" does not exist  
handling 20200127205541-init_db.sql
```

```
simple=# select * from migrations;
```

id	applied_at
20200127205541-init_db.sql	2020-02-02 15:59:13.596424+00

(1 row)

# rubenv/sql-migrate

Ошибка в нумерации

20200127205541-init\_db.sql

**20200127205542-test.sql**

20200127205545-simple.sql

# rubenv/sql-migrate

Ошибка в нумерации

20200127205541-init\_db.sql

**20200127205542-test.sql**

20200127205545-simple.sql

```
$ sql-migrate up
```

```
> Applied 1 migration
```

# rubenv/sql-migrate

## Ошибка в нумерации

```
simple=# select * from migrations;
```

id		applied_at
-----+-----		
20200127205541-init_db.sql		2020-02-02 12:35:13.746002+00
20200127205545-simple.sql		2020-02-02 15:25:06.768433+00
20200127205542-test.sql		2020-02-02 16:15:15.652378+00

# rubenv/sql-migrate

## Ошибка в нумерации

```
simple=# select * from migrations;
```

id	applied_at
20200127205541-init_db.sql	2020-02-02 16:17:01.065176+00
20200127205542-test.sql	2020-02-02 16:17:01.078368+00
20200127205545-simple.sql	2020-02-02 16:17:01.091582+00

# rubenv/sql-migrate

## Ошибка в нумерации

```
simple=# select * from migrations;
```




id	applied_at
000001-init_db.sql	2020-02-02 16:20:34.911222+00
000002-test.sql	2020-02-02 16:20:34.923781+00
000002-simple.sql	2020-02-02 16:20:52.643662+00



# rubenv/sql-migrate

- можно использовать как cli инструмент или как библиотеку
- возможность читать миграции из различных источников
- только sql-файлы для определения миграций
- поддержка нескольких типов БД в одном проекте
- изменения применяются и отменяются атомарно
- нужно следить за нумерацией миграций

# pressly/goose

 Watch ▾	47	 Star	1.1k	 Fork	200
---	----	--	------	--	-----

- PostgreSQL, MySQL, SQLite 3, MS SQL, Redshift
- миграции определяются с помощью sql и go
- можно использовать как cli инструмент или go-библиотеку
- изменения применяются и отменяются атомарно
- доступные команды: create, up, down, redo, reset, status, version, fix

# pressly/goose

## Ошибка в запросе

```
-- 20200123112616_init_db.sql
```

```
-- +goose Up
```

```
create table product(...);
```

```
insert into product values (1, 'test', 1, 'test', 1);
```

```
insert into product values (1, 'test', 2, 'test', 2);
```

```
create table attribute(...);
```

```
-- +goose Down
```

```
...
```

# pressly/goose

## Ошибка в запросе

```
$ goose up
```

```
> goose run: failed to run SQL migration
```

```
"20200123112616_init_db.sql": failed to execute SQL  
query "insert into public.product values (1, 'test', 2,  
'test', 2);\n": pq: duplicate key value violates unique  
constraint "product_pkey"
```

# pressly/goose

## Ошибка в запросе

```
simple=# select * from goose_db_version;
```

id	version_id	is_applied	timestamp
1	0	t	2020-02-02 17:02:30.005199

# pressly/goose

Исправили ошибку

```
$ goose up
```

```
> OK      20200123112616_init_db.sql
```

```
goose: no migrations to run. current version:  
20200123112616
```

# pressly/goose

## Исправили ошибку

```
simple=# select * from goose_db_version;
```

id	version_id	is_applied	tstamp
1	0	t	2020-02-02 17:10:16.393046
2	20200123112616	t	2020-02-02 17:10:48.822379

# pressly/goose

Ошибка в нумерации

20200123112616\_init\_db.sql

**20200123112617\_create\_something.sql**

20200123112618\_create\_index.sql



# pressly/goose

Ошибка в нумерации

20200123112616\_init\_db.sql

**20200123112617\_create\_something.sql**

20200123112618\_create\_index.sql

```
$ goose up
```

```
> goose: no migrations to run. current version:
```

```
20200123112618
```

# pressly/goose

Ошибка в нумерации

20200123112616\_init\_db.sql

**20200123112617\_create\_something.sql**

20200123112617\_create\_index.sql

# pressly/goose

## Ошибка в нумерации

```
$ goose up
```

```
> panic: goose: duplicate version 20200123112617
```

```
detected:
```

```
migration/migrations/20200123112617_create_something.sql
```

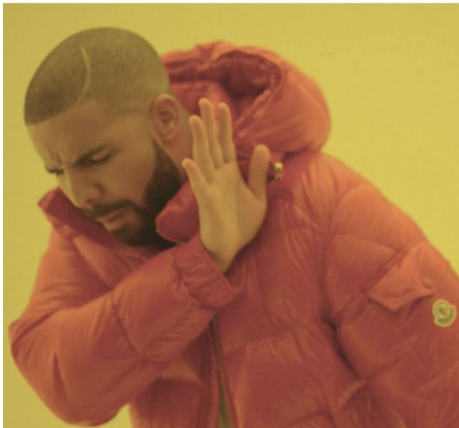
```
migration/migrations/20200123112617_create_index.sql
```

```
goroutine 1 [running]:
```

```
github.com/pressly/goose.Migrations.Less(0xc00004a0c0,  
0x2, 0x2, 0x1, 0x0, 0xc0000ebf90)
```

```
...
```

# pressly/goose



```
20200123112616_init_db.sql  
20200123112617_create_something.sql  
20200123112618_create_index.sql
```



```
00001_init_db.sql  
00002_create_index.sql  
00003_create_something.sql
```

# pressly/goose

Ошибка в запросе для отката

```
-- 20200123112616_init_db.sql
```

```
...
```

```
-- +goose Down
```

```
drop table product;
```

```
drop table attributes;
```

# pressly/goose

Ошибка в запросе для отката

```
$ goose down
```

```
> goose run: failed to run SQL migration
```

```
"20200123112616_init_db.sql": failed to execute SQL  
query "drop table public.attributes;\n": pq: table  
"attributes" does not exist
```

# pressly/goose

## Ошибка в запросе для отката

```
simple=# select * from goose_db_version;
```

id	version_id	is_applied	tstamp
1	0	t	2020-02-02 17:10:16.393046
2	20200123112616	t	2020-02-02 17:10:48.822379

# pressly/goose

Ошибка в запросе для отката

Schema	Name	Type	Owner
public	attribute	table	test
public	goose_db_version	table	test
public	product	table	test



# pressly/goose

- go и sql миграции
- можно использовать как cli инструмент или как библиотеку
- изменения применяются и отменяются атомарно
- нужно быть осторожнее с временной нумерацией версий

# Пример реализации

migration/migrations/00001\_init\_db.sql

```
-- +goose Up
```

```
create table product(...);
```

```
insert into product values (1, 'test', 1, 'test', 1);
```

```
insert into product values (2, 'test', 2, 'test', 2);
```

```
create table attribute(...);
```

```
-- +goose Down
```

```
drop table attribute;
```

```
drop table product;
```

# Пример реализации

migration/migrations/00002\_add\_test\_table.go

```
package migrations
```

```
import "github.com/pressly/goose"
```

```
func init() {  
    goose.AddMigration(Up00002, Down00002)  
}
```

# Пример реализации

migration/migrations/00002\_add\_test\_table.go

```
func Up00002(tx *sql.Tx) error {  
    query := `create table test (...);`  
    _, err := tx.Exec(query)  
    if err != nil {  
        return err  
    }  
    return nil  
}
```

# Пример реализации

migration/main.go

```
command := flag.String("c", "status", "command")  
dir := flag.String("dir", "migration/migrations", "mgt dir")  
flag.Parse()
```

# Пример реализации

migration/main.go

```
command := flag.String("c", "status", "command")
dir := flag.String("dir", "migration/migrations", "mgt dir")
flag.Parse()
```

```
dsn := "postgres://test:111@localhost:5432"
db, err := sql.Open("postgres", dsn)
if err != nil {
    log.Fatalf("-dbstring=%q: %v\n", dsn, err)
}
```

# Пример реализации

migration/main.go

```
if err := goose.SetDialect("postgres"); err != nil {  
    log.Fatal(err)  
}
```

```
if err := goose.Run(*command, db, *dir); err != nil {  
    log.Fatalf("goose run: %v", err)  
}
```

# Пример реализации

migration/main.go

```
package main
```

```
import (
```

```
    ...
```

```
    _ "simple/migration/migrations"
```

```
)
```



# Пример реализации

## Dockerfile

```
FROM golang:1.13.6-alpine AS goose
```

```
ENV CGO_ENABLED=0
```

```
RUN apk update
```

```
RUN apk add --no-cache git gcc
```

```
RUN go get -u github.com/pressly/goose/cmd/goose
```

# Пример реализации

## Dockerfile

```
FROM golang:1.13.6-alpine AS build
```

```
RUN mkdir /app
```

```
ADD . /app/
```

```
WORKDIR /app
```

```
RUN go build -o /bin/simple cmd/simple/main.go
```

```
RUN go build -o /bin/migrate migration/main.go
```

# Пример реализации

## Dockerfile

```
FROM alpine:latest
COPY --from=build /bin/simple /bin/simple
COPY --from=build /bin/migrate /bin/migrate
COPY --from=goose /go/bin/goose /usr/bin/goose
COPY migration/migrations migration_db

CMD ["/bin/simple"]
```

# Полезные ссылки

Рассмотренные инструменты:

- <https://github.com/pressly/goose>
- <https://github.com/rubenv/sql-migrate>
- <https://github.com/golang-migrate/migrate>

Инструменты, библиотеки, фреймворки на Go:

<https://awesome-go.com/>