



# Go-Lang Database Migration

Eko Kurniawan Khannedy

# Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 11+ years experiences
- [www.programmerzamannow.com](http://www.programmerzamannow.com)
- [youtube.com/c/ProgrammerZamanNow](https://youtube.com/c/ProgrammerZamanNow)





# Eko Kurniawan Khannedy

- Telegram : [@khannedy](https://t.me/khannedy)
- Facebook : [fb.com/ProgrammerZamanNow](https://fb.com/ProgrammerZamanNow)
- Instagram : [instagram.com/programmerzamannow](https://instagram.com/programmerzamannow)
- Youtube : [youtube.com/c/ProgrammerZamanNow](https://youtube.com/c/ProgrammerZamanNow)
- Telegram Channel : [t.me/ProgrammerZamanNow](https://t.me/ProgrammerZamanNow)
- Email : [echo.khannedy@gmail.com](mailto:echo.khannedy@gmail.com)



# Sebelum Belajar

- MySQL Database
- Golang Database



# Agenda

- Pengenalan Database Migration
- Pengenalan Golang Migrate
- Menginstall Golang Migrate
- Membuat File Migration
- Menjalankan Migration
- Rollback Migration

---

# Pengenalan Database Migration



# Pengenalan Database Migration

- Saat ini, kebanyakan aplikasi yang dibuat akan membutuhkan database
- Saat aplikasi berjalan, biasanya database sudah siap digunakan, artinya table, kolom dan semua relasinya sudah dibuat di awal sebelum aplikasi dijalankan
- Apa yang terjadi ketika misal pada saat aplikasi sudah berjalan, kita perlu menambah fitur baru, lalu butuh mengubah struktur table di database?
- Biasanya kita akan mengubahnya di database langsung, lalu melakukan perubahan kode program
- Hal ini mungkin terlihat sederhana, namun ketika skalanya sudah besar, dan anggota tim sudah banyak, maka perubahan langsung ke database bukanlah hal sederhana lagi
- Kita harus bisa melakukan tracking apa saja yang berubah, dan memastikan semua anggota tim tahu perubahannya, sehingga bisa dilakukan hal yang sama di komputer masing-masing



# Keuntungan Database Migration

- Oleh karena itu, Database Migration sangat diperlukan
- Database Migration adalah mekanisme untuk melakukan tracking perubahan struktur database, dari mulai awal dibuat sampai perubahan terakhir yang dilakukan
- Mirip seperti Git, dimana melakukan tracking semua perubahan kode program
- Dengan menggunakan Database Migration, semua tim member bisa melihat perubahan struktur database, dan bisa dengan mudah menjalankan perubahan tersebut di tiap komputer masing-masing
- Selain itu, dengan adanya Database Migration, kita bisa melakukan review terlebih dahulu, sebelum menjalankan perubahan di database, jaga-jaga ada perubahan yang salah, yang bisa berdampak berbahaya ke database



---

# Pengenalan Golang Migrate



# Pengenalan Golang Migrate

- Golang Migrate adalah salah satu tool untuk Database Migration yang populer digunakan oleh programmer Golang
- Golang Migrate bisa diintegrasikan dengan aplikasi, atau dijalankan sebagai aplikasi standalone
- Golang Migrate mendukung banyak sekali database, seperti MySQL, PostgreSQL, Sqlite, MongoDB, Cassandra, dan lain-lain
- <https://github.com/golang-migrate/migrate>

---

# Menginstall Golang Migrate



# Menginstall Golang Migrate

- Untuk menginstall Golang Migrate, sangat mudah, kita bisa gunakan perintah berikut :  
`go install -tags 'database1,database2' github.com/golang-migrate/migrate/v4/cmd/migrate@latest`
- Sesuaikan dengan database yang ingin kita gunakan, bisa lebih dari satu dengan cara menambahkan koma



## Kode : Menginstall Golang Migrate

```
→ ~ go install -tags 'postgres,mysql' github.com/golang-migrate/migrate/v4/cmd/migrate@latest
→ ~ cd $GOPATH
→ go cd bin
→ bin ls -l
total 40664
-rwxr-xr-x 1 khannedy staff 7844272 Nov 27 2021 belajar-golang-restful-api
-rwxr-xr-x 1 khannedy staff 6996224 Sep 21 17:25 migrate
-rwxr-xr-x 1 khannedy staff 5967968 Nov 27 2021 wire
→ bin █
```



# Aplikasi migrate

- Saat menginstall Golang Migrate, secara otomatis terdapat executable file di folder \$GOPATH/bin/ dengan nama migrate
- File migrate tersebut adalah aplikasi Golang Migrate yang akan kita gunakan untuk membuat Database Migration

# Kode : Aplikasi migrate

```
➔ bin migrate
Usage: migrate OPTIONS COMMAND [arg...]
       migrate [ -version | -help ]

Options:
  -source          Location of the migrations (driver://url)
  -path            Shorthand for -source=file://path
  -database        Run migrations against this database (driver://url)
  -prefetch N      Number of migrations to load in advance before executing (default 10)
  -lock-timeout N  Allow N seconds to acquire database lock (default 15)
  -verbose         Print verbose logging
  -version         Print version
  -help           Print usage

Commands:
  create [-ext E] [-dir D] [-seq] [-digits N] [-format] [-tz] NAME
    Create a set of timestamped up/down migrations titled NAME, in directory D with extension E.
    Use -seq option to generate sequential up/down migrations with N digits.
    Use -format option to specify a Go time format string. Note: migrations with the same time cause "duplicate migration version" error.
    Use -tz option to specify the timezone that will be used when generating non-sequential migrations (defaults: UTC).

  goto V          Migrate to version V
  up [N]          Apply all or N up migrations
  down [N] [-all] Apply all or N down migrations
```

---

# Membuat Project





# Membuat Project

- Sebagai contoh, kita akan menggunakan project Golang RESTful API yang pernah kita buat, lalu kita akan tambahkan Database Migration ke project tersebut
- <https://github.com/ProgrammerZamanNow/belajar-golang-restful-api>

---

# Membuat Database Migration



# Membuat Database Migration

- Untuk membuat database migration, kita bisa gunakan perintah :  
migrate create -ext sql -dir db/migrations nama\_file\_migration
- -ext adalah file extension, artinya kita membuat file .sql
- -dir adalah folder tempat disimpan
- Usahakan tidak menggunakan spasi pada nama file migration



## Kode : Membuat Database Migration

```
master) ✕ migrate create -ext sql -dir db/migrations create_table_category  
s/belajar-golang-database-migration/db/migrations/20220921103313_create_table_category.up.sql  
s/belajar-golang-database-migration/db/migrations/20220921103313_create_table_category.down.sql  
master) ✕
```



# File Migration

- File migration akan diawali dengan waktu ketika kita membuat file migration, lalu diikuti dengan nama migration, dan diakhiri dengan tipe migration
- misal 20220921103313\_create\_table\_category.up.sql
- Kenapa diawali dengan waktu? Agar file migration selalu berurut sesuai waktu kita membuat file migration tersebut

---

# Migration Up



# Migration Up

- Saat kita membuat file database migration, file dengan akhiran up adalah file yang harus kita isi dengan perubahan yang ingin kita tambahkan
- Misal, sekarang kita akan tambahkan table category, sesuai dengan aplikasi RESTful API yang sudah kita buat

## Kode : Migration Up

SQL 20220921103313\_create\_table\_category.up.sql

No data sources are configured to run this SQL and provide advanced code assistance.

```
1 CREATE TABLE category
2 (
3     id INT NOT NULL AUTO_INCREMENT,
4     name VARCHAR(255) NOT NULL,
5     PRIMARY KEY (id)
6 ) ENGINE = InnoDB;
```



---

# Migration Down



# Migration Down

- Setiap file migration, selain file up, terdapat juga file down
- File down ini adalah file yang berisikan kode untuk mengembalikan perubahan yang kita lakukan di file up
- Kenapa ini diperlukan? Karena misal terjadi masalah di aplikasi, namun database migration terlanjur dijalankan, kita bisa melakukan rollback dengan cara menjalankan file down, karena berisikan kode untuk mengembalikan perubahan di file up
- Pada kasus ini, misal kita akan menghapus lagi table category



## Kode : Migration Down

SQL 20220921103313\_create\_table\_category.down.sql ×

No data sources are configured to run this SQL and provide advanced code assistance.

```
1 DROP TABLE IF EXISTS category;
```

---

# Membuat Database



# Membuat Database

- Sebelum menjalankan Database Migration, sekarang kita perlu membuat dulu database nya
- Hal ini karena pembuatan database tidak dilakukan di database migration, biasanya dilakukan manual diawal
- Pada kasus ini, kita menggunakan database mysql, dan kita perlu ubah juga kode koneksi database di aplikasi agar terhubung dengan database baru

# Kode : Golang Connection

```
database.go x
1  package app
2
3  import ...
4
5
6
7
8
9  func NewDB() *sql.DB {
10     db, err := sql.Open("mysql", "root@tcp(localhost:3306)/belajar_golang_database_migration")
11     helper.PanicIfError(err)
12
13     db.SetMaxIdleConns(5)
14     db.SetMaxOpenConns(20)
15     db.SetConnMaxLifetime(60 * time.Minute)
16     db.SetConnMaxIdleTime(10 * time.Minute)
17 }
```



## Kode : Membuat Database

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> CREATE DATABASE belajar_golang_database_migration;
```

```
Query OK, 1 row affected (0.09 sec)
```

```
mysql>
```

---

# Menjalankan Migration





# Menjalankan Migration

- Selanjutnya, setelah selesai, kita bisa menjalankan database migration menggunakan perintah :  
migrate -database "koneksidatabase" -path folder up
- -database harus berisikan koneksi database, misal untuk mysql, bisa menggunakan :  
mysql://user:password@tcp(host:port)/nama\_database
- Untuk database lainnya, bisa lihat di halaman dokumentasinya :  
<https://github.com/golang-migrate/migrate#databases>
- -path adalah lokasi folder file database migration
- up adalah perintah untuk menjalankan database migration dengan mode up



## Kode : Menjalankan Migration

```
aster) x  
aster) x migrate -database "mysql://root@tcp(localhost:3306)/belajar_golang_database_migration" -path db/migrations up  
309912ms)  
aster) x
```

---

# Migration State



# Migration State

- Saat kita sudah melakukan migration, lalu kita menambah file migration baru, apa yang terjadi jika kita menjalankan migration lagi?
- Golang Migrate akan menyimpan state terakhir kita menjalankan database migration, artinya tidak akan dijalankan dari awal lagi, melainkan dari file terakhir yang sukses di migrasi
- Jadi kita tidak perlu takut file akan dijalankan lagi, jadi tidak perlu dihapus file migration lama-nya
- Semua informasi state tersebut disimpan dalam table `schema_migrations`



## Kode : Schema Migrations

```
mysql> show tables;
+-----+
| Tables_in_belajar_golang_database_migration |
+-----+
| category                                     |
| schema_migrations                           |
+-----+
2 rows in set (0.00 sec)

mysql> select * from schema_migrations;
+-----+-----+
| version      | dirty |
+-----+-----+
| 20220921103313 | 0     |
+-----+-----+
1 row in set (0.00 sec)
```

---

# Rollback Migration



# Rollback Migration

- Pada waktu misal terjadi masalah pada aplikasi, yang menyebabkan kita harus melakukan rollback perubahan, apa yang kita harus lakukan?
- Fitur itu sudah ada di Golang Migrate, jadi kita bisa menjalankan mode down untuk melakukan rollback dengan perintah :  
`migrate -database "koneksidatabase" -path folder down`



## Kode : Migration Down

```
migrate -database "mysql://root@tcp(localhost:3306)/belajar_golang_database_migration" -path db/migrations down  
[y/N]
```

```
→ belajar-golang-database-migration git:(master) ✕ migrate
```

```
Are you sure you want to apply all down migrations? [y/N]
```

```
y
```

```
Applying all down migrations
```

```
20220921103313/d create_table_category (69.114489ms)
```

```
→ belajar-golang-database-migration git:(master) ✕
```





## Kode : Schema Migration

```
mysql> show tables;
+-----+
| Tables_in_belajar_golang_database_migration |
+-----+
| schema_migrations                          |
+-----+
1 row in set (0.00 sec)

mysql> select * from schema_migrations;
Empty set (0.00 sec)

mysql> █
```

---

# Migrasi ke Versi Tertentu



## Migrasi ke Versi Tertentu

- Saat menggunakan mode up atau down, secara otomatis akan melakukan migrasi seluruh file
- Kadang pada kenyataanya, kita mungkin hanya ingin melakukan up atau down ke versi tertentu saja
- Misal jika ingin melakukan rollback, mungkin kita hanya ingin rollback satu versi saja, tidak mau melakukan rollback semua versi
- Untuk kasus ini, setelah perintah up atau down, kita bisa memasukkan angka, yaitu jumlah migration yang ingin kita eksekusi



# Tugas

- Buatlah 3 migration file baru



## Kode : Membuat Migration

```
ter) ✕ migrate create -ext sql -dir db/migrations create_table_first  
elajar-golang-database-migration/db/migrations/20220921110749_create_table_first.up.sql  
elajar-golang-database-migration/db/migrations/20220921110749_create_table_first.down.sql  
ter) ✕ migrate create -ext sql -dir db/migrations create_table_second  
elajar-golang-database-migration/db/migrations/20220921110752_create_table_second.up.sql  
elajar-golang-database-migration/db/migrations/20220921110752_create_table_second.down.sql  
ter) ✕ migrate create -ext sql -dir db/migrations create_table_third  
elajar-golang-database-migration/db/migrations/20220921110755_create_table_third.up.sql  
elajar-golang-database-migration/db/migrations/20220921110755_create_table_third.down.sql  
ter) ✕
```



# Tugas

- Lakukan up
- Lakukan down
- Lakukan up beberapa versi
- Lakukan down beberapa versi

---

# Dirty State

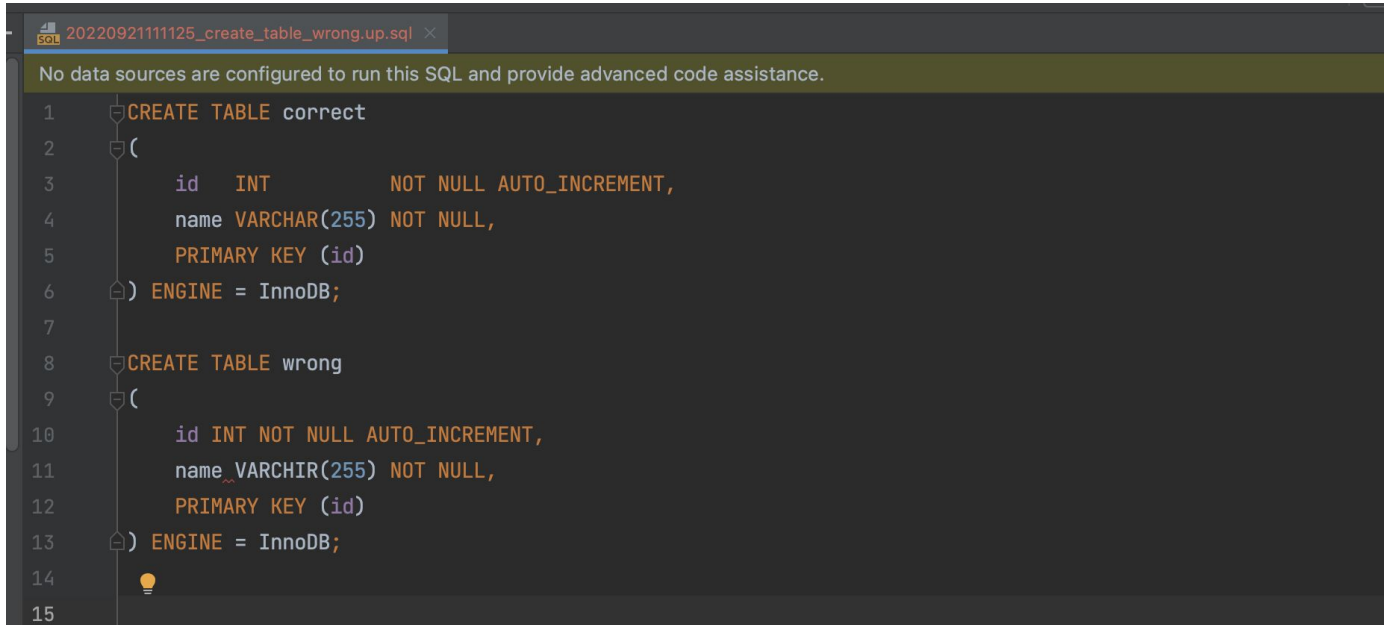


# Dirty State

- Saat kita membuat database migration, kadang kesalahan sering terjadi
- Misal saja, kita melakukan typo sehingga membuat perintah SQL nya salah
- Jika kita terlanjur menjalankan database migration, maka state akan berubah menjadi Dirty State
- State gimana kita tidak bisa melakukan up atau down lagi
- Pada kasus ini, kita harus perbaiki manual, kenapa harus manual? Karena tidak ada cara otomatis memperbaikinya



# Kode : Migration Error



```
20220921111125_create_table_wrong.up.sql x
No data sources are configured to run this SQL and provide advanced code assistance.

1 CREATE TABLE correct
2 (
3     id INT NOT NULL AUTO_INCREMENT,
4     name VARCHAR(255) NOT NULL,
5     PRIMARY KEY (id)
6 ) ENGINE = InnoDB;
7
8 CREATE TABLE wrong
9 (
10    id INT NOT NULL AUTO_INCREMENT,
11    name_VARCHIR(255) NOT NULL,
12    PRIMARY KEY (id)
13 ) ENGINE = InnoDB;
14
15
```



# Kode : Menjalankan Migration

```
Terminal: Local x + v
➔ belajar-golang-database-migration git:(master) ✕ migrate -database "mysql://root@tcp(localhost:3306)/belajar_golang_database_migration" -path db/migrations up
error: migration failed in line 0: CREATE TABLE correct
(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
) ENGINE = InnoDB;

CREATE TABLE wrong
(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHIR(255) NOT NULL,
    PRIMARY KEY (id)
) ENGINE = InnoDB;

(details: Error 1064: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'VARCHIR(255) NOT NULL,
PRIMARY KEY (id)
) ENGINE = InnoDB' at line 4)
➔ belajar-golang-database-migration git:(master) ✕
```



# Permasalahan

- Permasalahan migration ini adalah, kita membuat dua table di file migration, pada pembuatan table pertama sukses, namun pada table kedua gagal
- Artinya file migration tidak sempurna, dan kita juga tidak bisa melakukan rollback, karena table kedua belum sukses dibuat
- Pada kondisi ini, terjadi yang namanya Dirty State, dimana kita tidak bisa melakukan up atau down, yang perlu kita lakukan adalah memperbaiki secara manual



## Kode : Schema Migrations

```
mysql> show tables;
```

```
+-----+  
| Tables_in_belajar_golang_database_migration |  
+-----+  
| category                                |  
| correct                                |  
| first                                  |  
| schema_migrations                      |  
| second                                |  
| third                                  |  
+-----+
```

```
6 rows in set (0.00 sec)
```

```
mysql> select * from schema_migrations;
```

```
+-----+-----+  
| version      | dirty |  
+-----+-----+  
| 20220921111125 | 1     |  
+-----+-----+
```

```
1 row in set (0.00 sec)
```



## Kode : Perbaiki Manual

```
mysql> drop table correct;  
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> █
```



# Mengubah Versi

- Setelah kita memperbaiki secara manual, selanjutnya kita perlu mengubah versi migration di table `schema_migrations`
- Kita bisa lakukan manual, atau bisa otomatis menggunakan perintah :  
`migrate -database "koneksi_database" -path folder force versi`
- Dimana versi adalah versi dari file database migration
- Pada kasus ini, kita akan gunakan satu versi sebelum migration yang gagal



## Kode : Mengubah Versi Migration

```
✗ migrate -database "mysql://root@tcp(localhost:3306)/belajar_golang_database_migration" -path db/migrations force 20220921110755
✗ migrate -database "mysql://root@tcp(localhost:3306)/belajar_golang_database_migration" -path db/migrations version
```



## Selanjutnya

- Selanjutnya kita bisa perbaiki file database migration nya
- Lalu ulangi jalankan file migration nya



---

# Mencoba Aplikasi



# Mencoba Aplikasi

- Sekarang sebelum kita menjalankan aplikasi, selalu jalankan database migration terlebih dahulu
- Dan selanjutnya kita bisa mencoba menjalankan aplikasinya

---

# Materi Selanjutnya



# Materi Selanjutnya

- Belajar Framework dan Library Go-Lang
- Studi Kasus Membuat Aplikasi menggunakan Go-Lang