Database Migrations: Multi-Dimensional Versioning

İbrahim Serdar Açıkgöz

23.11.2024

About me

Persons

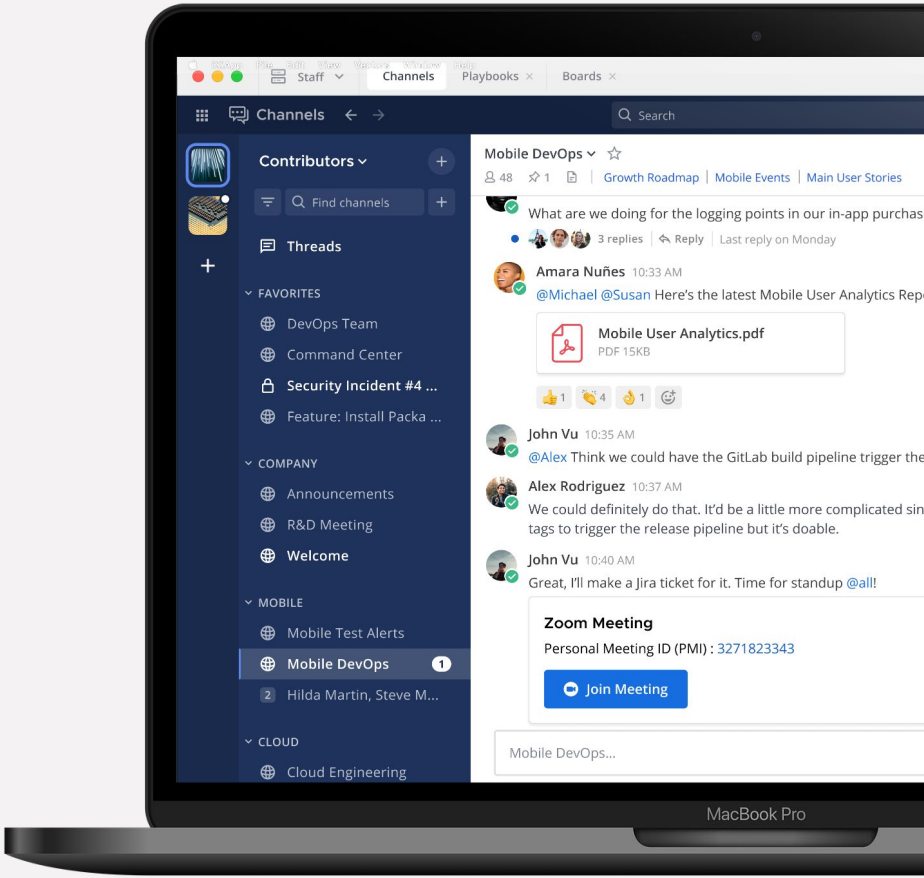
Id bigint 🔑	CreateAt bigint	FullName text	Occupation text	Social json
**********	529318812	İbrahim Serdar Açıkgöz	Software Engineer	{“x”:“isacikgoz”}

Outline

- Problem domain
- Approaches, alternatives
- Proposed solution

Mattermost

- Open Source Collaboration Software
- Highly Customizable and Secure
- Flexible and Developer Friendly



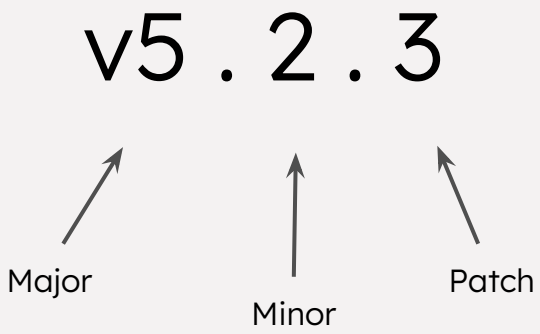
On prem releases

Releases

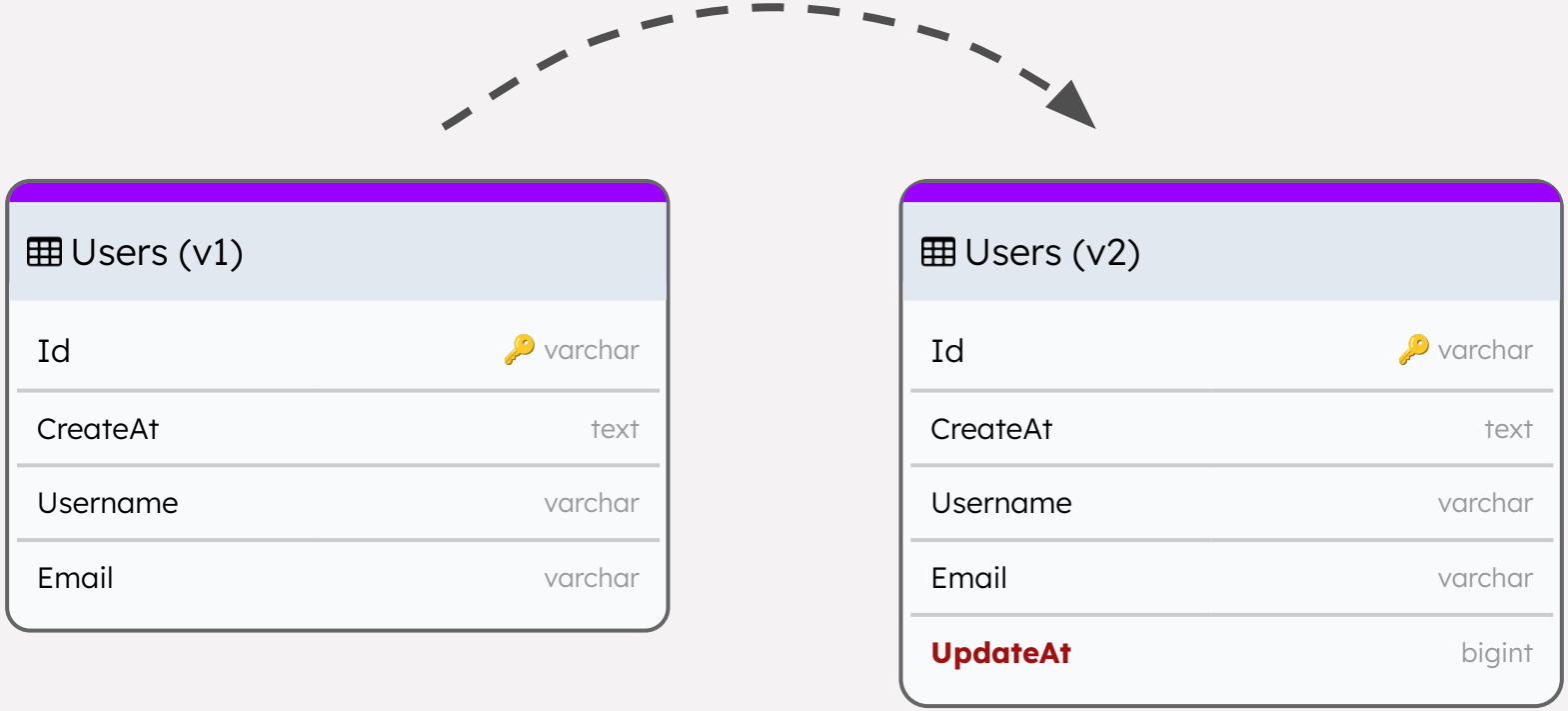
Release	Released On	Support Ends	Extended Support
v10.2.0	2024-11-15	2025-02-15	-
v10.1.3	2024-10-16	2025-01-15	-
v10.0.2	2024-09-16	2024-12-15	-
v9.11.4	2024-08-16	2025-05-15	Yes
v9.10.3	2024-07-16	2024-10-15	-
v9.9.3	2024-06-14	2024-09-15	-
v9.5.9	2024-02-16	2024-11-15	Yes

Versions

- Semantic versioning
- Sequence based



Schema migration



Applied migrations

db_migrations

Version	Name
128	create_scheduled_posts
127	add_mfa_used_ts_to_users
126	sharedchannels_remotes_add_deleteat
125	remoteclusters_add_default_team_id
124	remove_manage_team_permission
123	remove_upload_file_permission
122	preferences_value_length

Schema versions

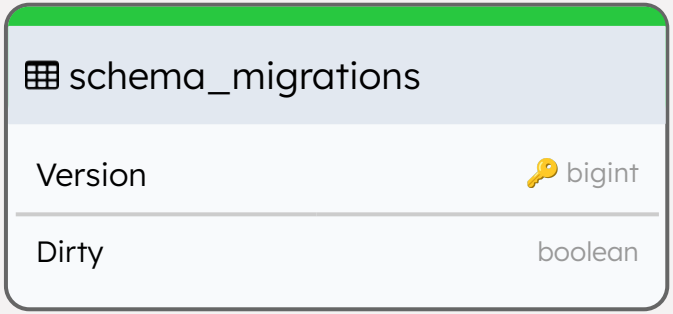
db_migrations (v9.5.11)

version bigint 🔑	name text
122	preferences_va…
121	remove_true_up…
120	create_channel…
...	...

db_migrations (v10.3.0)

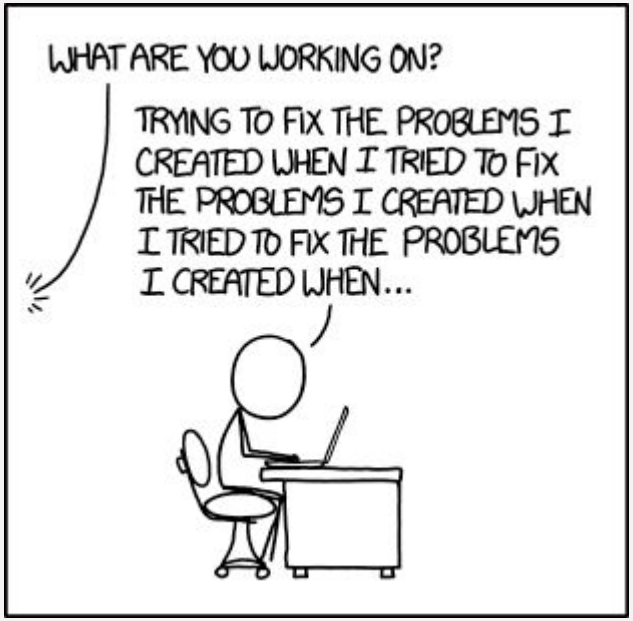
version bigint 🔑	name text
128	create_schedu…
127	add_mfa_used…
126	sharedchannel…
...	...

Versions



Rollback

- Errors during migrations?
- Dirty state, who is going to fix?

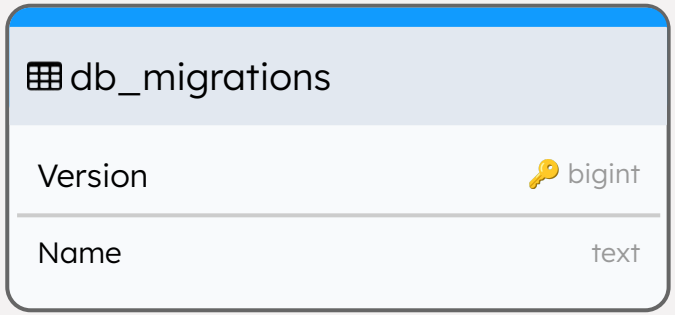


Source: https://xkcd.com/1739/

morph: new dimension

morph

- Schema migration tool
- Available both as CLI and Library
- Schema version as a table



Versions

db_migrations

version bigint 🔑	name text
128	create_scheduled_posts
127	add_mfa_used_ts_to_users
126	sharedchannels_remotes_add_deleteat
125	remoteclusters_add_default_team_id
124	remove_manage_team_permission
123	remove_upload_file_permission
122	preferences_value_length

Schema versions

db_migrations (v9.5.11)

version bigint 🔑	name text
122	preferences_va…
121	remove_true_up…
120	create_channel…
...	...

db_migrations (v9.5.12)

version bigint 🔑	name text
128	create_schedu…
122	preferences_va…
121	remove_true_up…
...	...

Schema versions

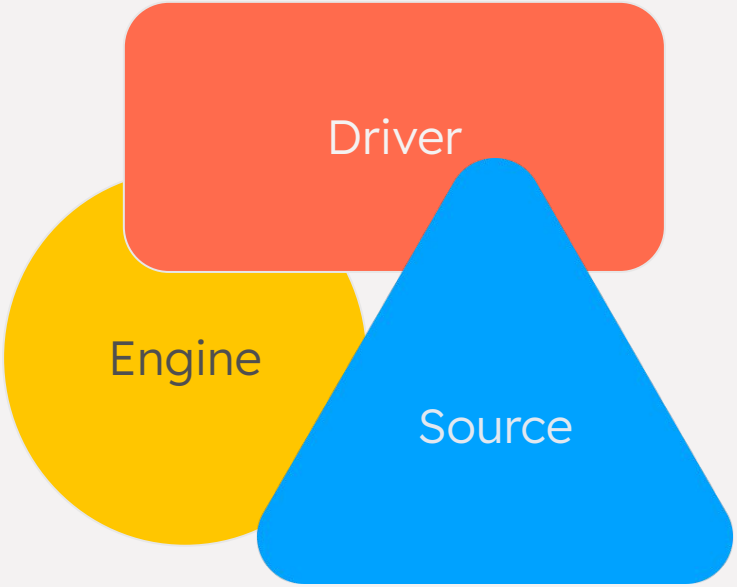
db_migrations (v9.5.12)

version bigint 🔑	name text
128	create_schedu…
122	preferences_va…
121	remove_true_up…
...	...

db_migrations (v10.3.0)

version bigint 🔑	name text
128	create_schedu…
127	add_mfa_used…
126	sharedchannel…
...	...

morph: components



```
import (
    "context"

    "github.com/mattermost/morph"
    ...
)

func migrate() error {
     ...
     engine, err := morph.New(context.TODO(), driver, src)
     if err != nil {
          return err
     }
     defer engine.Close()

     if err = engine.ApplyAll(); err != nil {
          return err
     }
     ...
}
```

morph: interfaces

```
type Driver interface {
     Ping() error
     Close() error
     Apply(migration *models.Migration, saveVersion bool) error
     AppliedMigrations() ([]*models.Migration, error)
     SetConfig(key string, value interface{}) error
}

type Source interface {
     Migrations() (migrations []*models.Migration)
}
```

morph: data types

```
type Direction string

const (
     Down Direction = "down"
     Up   Direction = "up"
)

type Migration struct {
     Bytes     []byte
     Name      string
     RawName   string
     Version   uint32
     Direction Direction
}
```

0000000001_create_user.up.sql
0000000001_create_user.down.sql

morph: Sources

- go: embed
- io.Reader (eg. os.File)
- text/template

```
type Source interface {
     Migrations() (migrations []*models.Migration)
}
```

morph: mutex

```
func New(ctx context.Context, driver drivers.Driver, source sources.Source, options
...EngineOption) (*Morph, error) {
	...
	if impl, ok := driver.(drivers.Lockable); ok && engine.config.LockKey != "" {
		mx, err := impl.NewMutex(engine.config.LockKey, engine.config.Logger)
		if err != nil {
			return nil, err
		}

		engine.mutex = mx
		err = mx.Lock(ctx)
		if err != nil {
			return nil, err
		}
	}
	...
}
```

morph: mutex

```
type Locker interface {
     // Lock refreshes the lock unless the context is canceled. If the mutex is already locked
     // by any other instance, including the current one, the calling goroutine blocks until the
     // mutex can be locked. Will return the error if the context is canceled.
     //
     // The mutex is locked only if a nil error is returned.
     Lock(ctx context.Context) error
     Unlock() error
}

type Lockable interface {
     NewMutex(key string, logger Logger) (Locker, error)
}
```

morph in action

- Embedded migrations (morph/sources/embedded)
- Support multiple database drivers (MySQL, Postgres) (morph/drivers/*)
- morph “plans”
- Interceptor callback

Recap

- On-prem distribution
- Supporting multiple DB vendors
- Error resilience & Recoverability

Thank you!