Mastering Neo4j With Go and GoGM

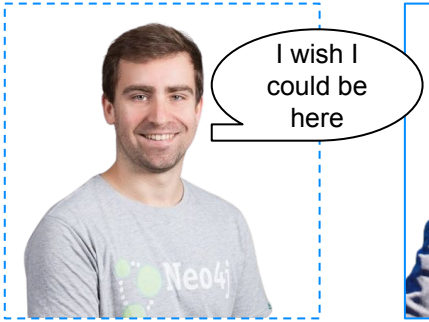
Eric Solender (@ctoeric97)
Florent Biville (@fbiville)
Nikita Wootten (@nettoowatikin)

GraphConnect 2022

MindStand neo4j

Presenters

Eric Solender
Mindstand
Chief Technical Officer
GoGM Maintainer



Florent Biville
Neo4j
Go Driver Maintainer

Nikita Wootten
Mindstand
Chief Data Scientist
GoGM Maintainer

MindStand neo4j

Plan

Workshop Plan:

1. Introduction to **Go**
2. Introduction to **Graph** Databases
3. Neo4j + **Go**
4. Neo4j + **GoGM**

Workshop Repository

MindStand neo4j

Workshop Prerequisites 📸

- Git: `git clone https://github.com/fbiville/graphconnect-go-workshop.git`
- Go 1.18 (or greater if you live in the future)
- Docker
- Visual Studio Code, Goland or <YourFavoriteEditor>

MindStand neo4j

Introduction to Go

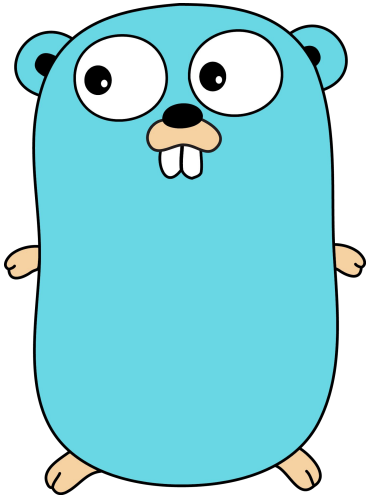
Introduction to Go

Created at Google in 2009

- Open-Source
- Compiled, Garbage Collected and
- Statically Typed

Strengths

- Fast Compile/Execution Time
- Approachable Concurrency

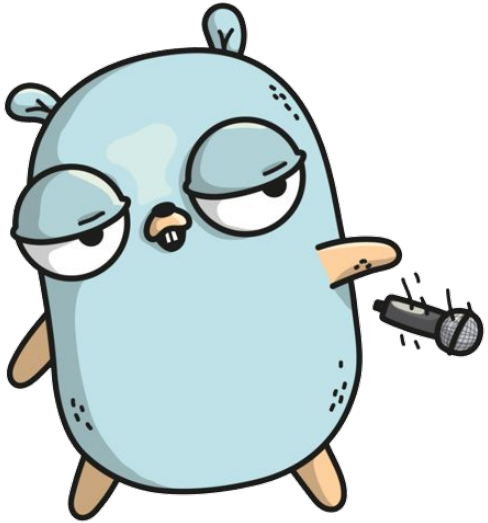


Introduction to Go - Examples

```go
package main

import "fmt"

func main() {
  message := "Hello GraphConnect!"
  // equivalent of:
  // var message string = "Hello GraphConnect!"
  fmt.Println(message)
}
```



Introduction to Go - Examples

```
type CustomerService struct {
  Repository CustomerRepository
}

type Customer struct {
  RegistrationId string
  Name           string
  Company        string
}

func (service *CustomerService) RegisterCustomer(customer Customer) (bool, error) {
  if customer.RegistrationId != "" {
     return false, nil
  }
  err := service.Repository.Register(customer)
  return err == nil, err
}
```

MindStand neo4j

Introduction to Go - Examples

```go
package main
import (
  "fmt"
  "math/rand"
  "sync"
  "time"
)
func main() {
  var semaphore sync.WaitGroup
  semaphore.Add(100)
  for i := 0; i < 100; i++ {
     go func(i int) {
        randomSleep()
        fmt.Println(i)
        semaphore.Done()
     }(i)
  }
  semaphore.Wait()
}
```

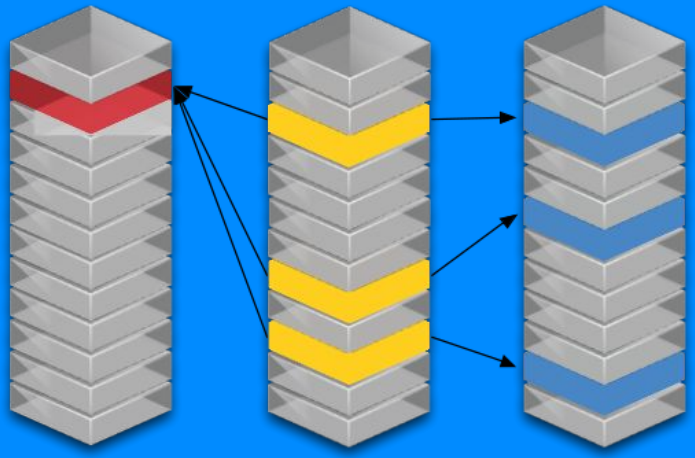
```go
func randomSleep() {
  amount := time.Duration(rand.Int31n(5))
  time.Sleep(amount * time.Second)
}
```

MindStand neo4j

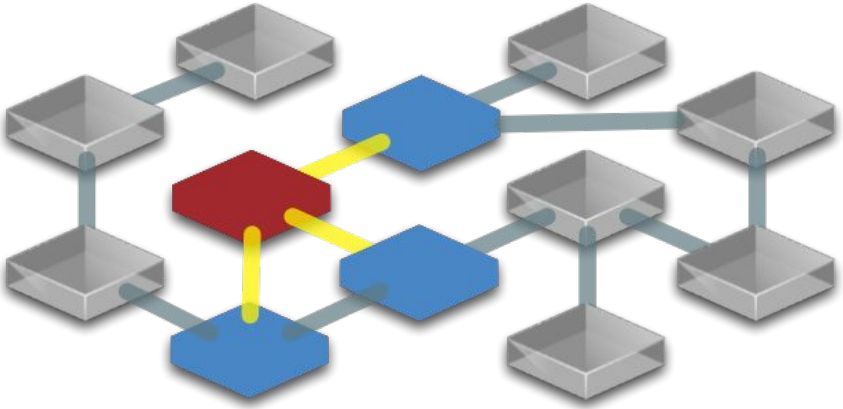
Workshop Part 1 Exercises: Introduction to Go

Introduction to Graph Databases

Relational Database

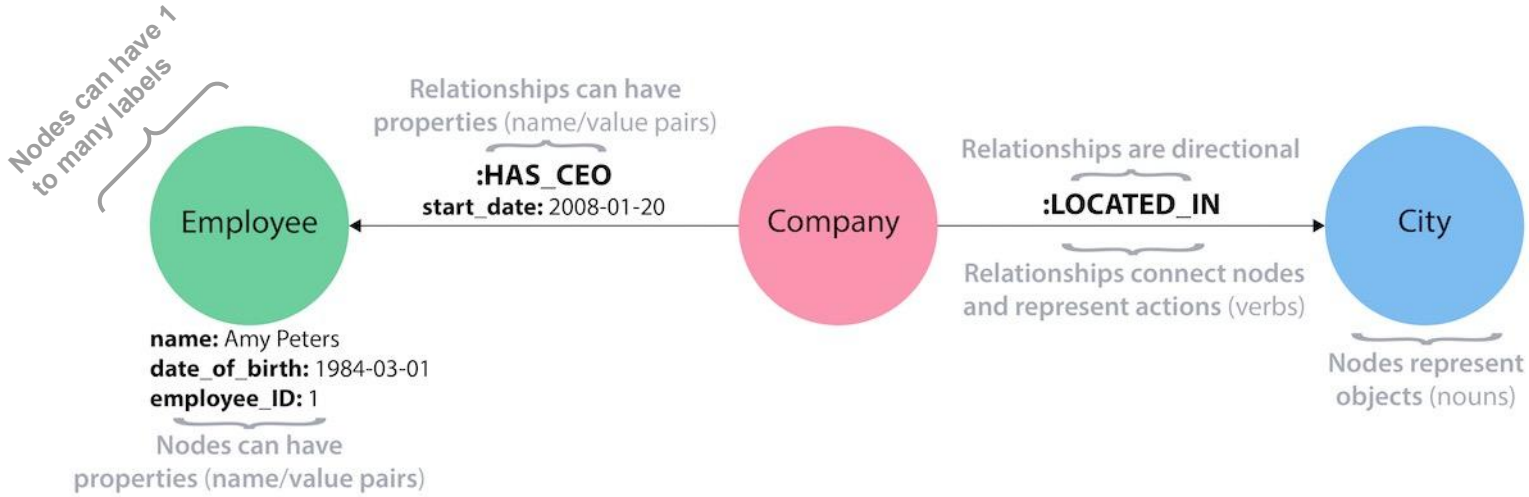


Graph Database



MindStand neo4j

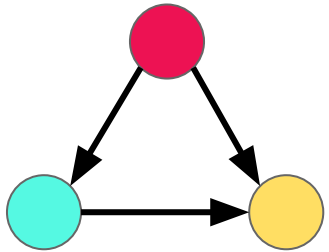
Property Graph



MindStand neo4j

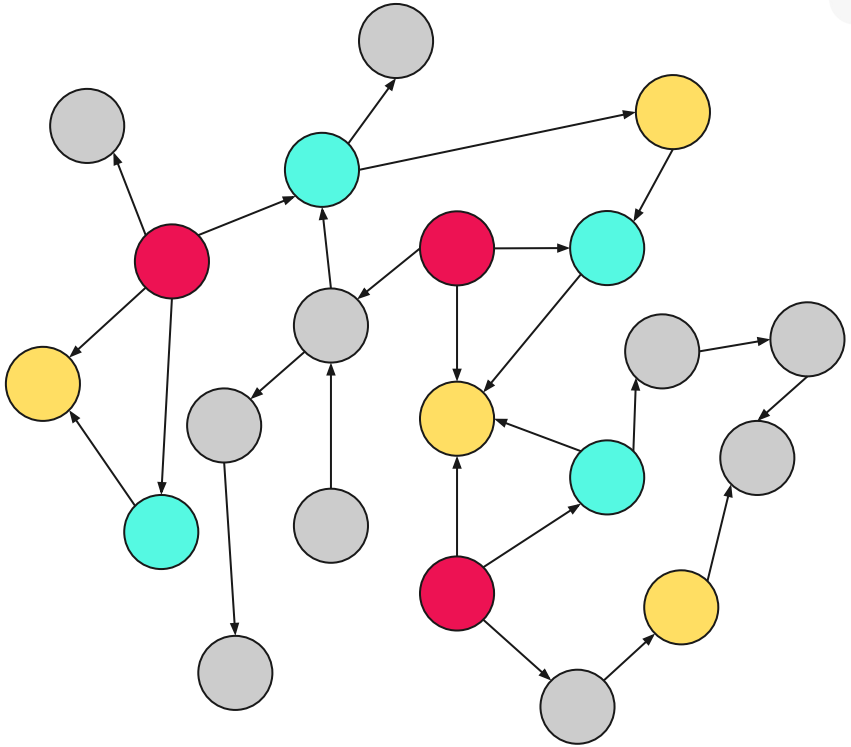
Graph Pattern Matching

Pattern to Find

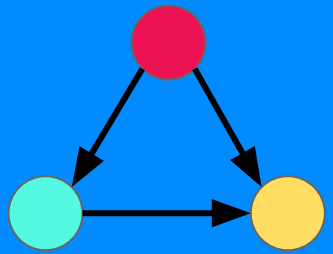


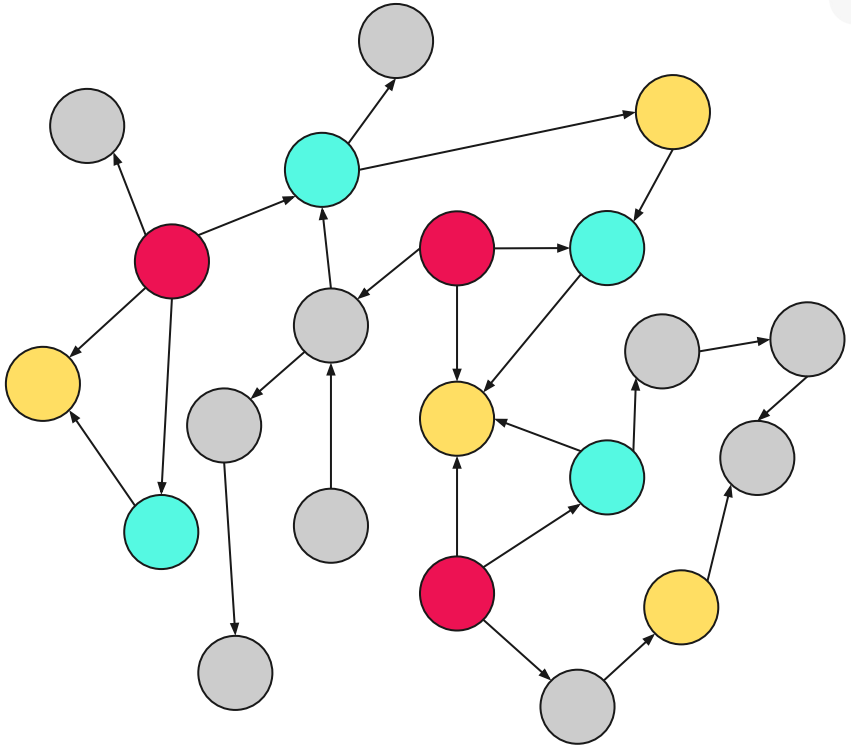
Graph Pattern Matching

Existing Data

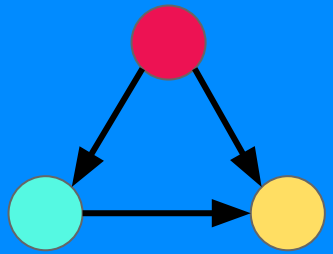


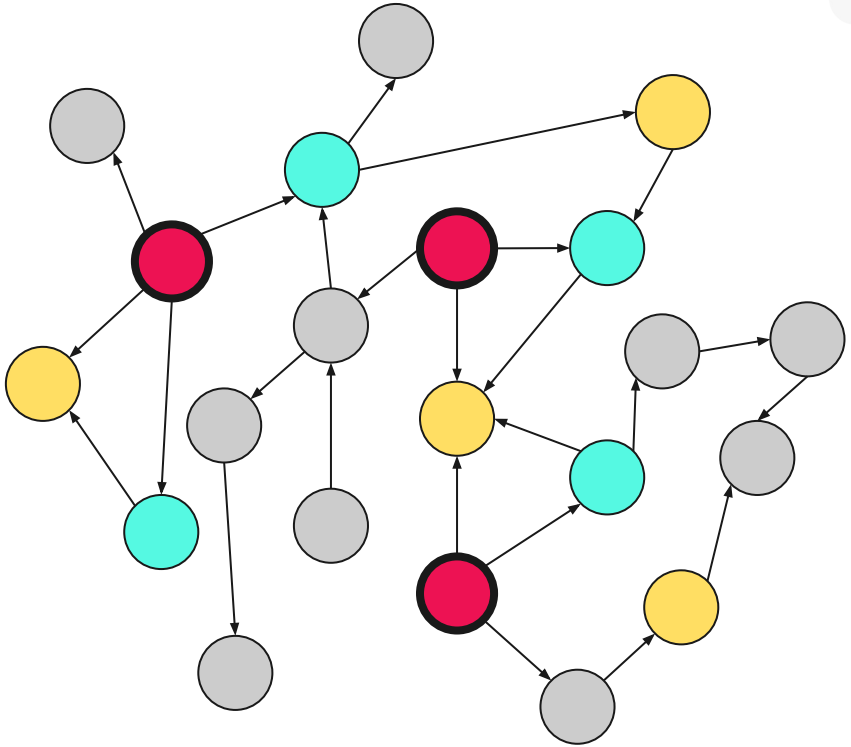
Graph Pattern Matching



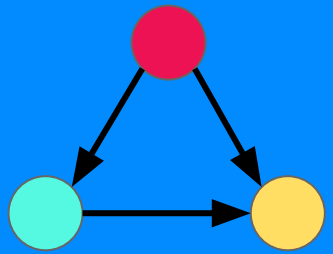


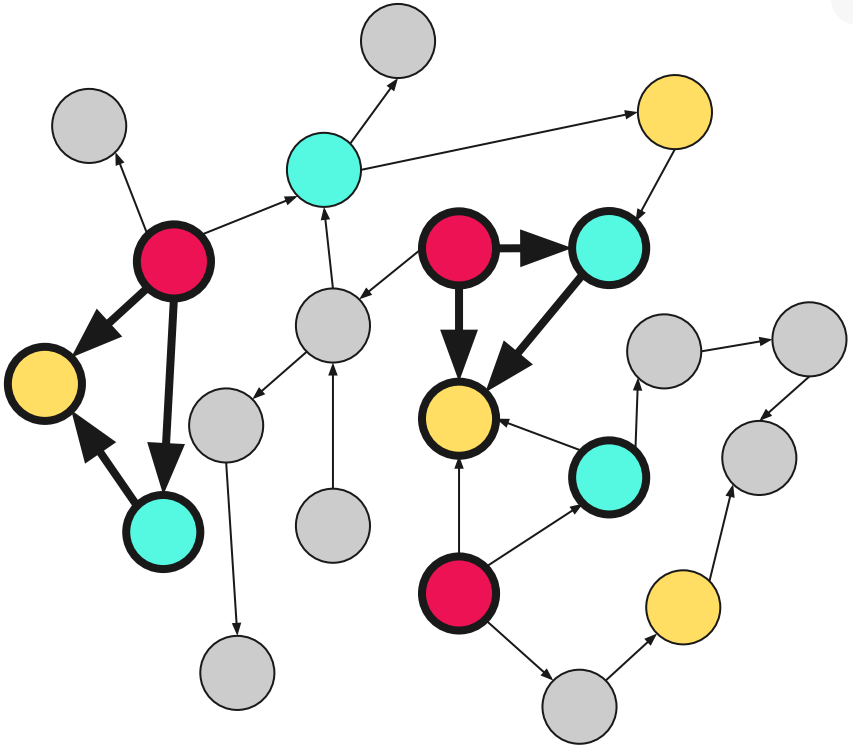
Graph Pattern Matching





Graph Pattern Matching





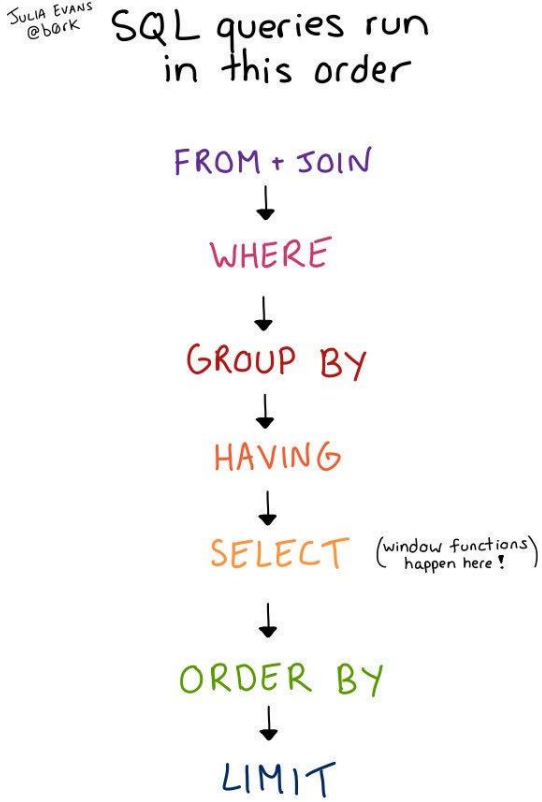
Graph Pattern Matching

What you describe(aka match) is what you get!

- The runtime infers the graph traversals for you.

MindStand neo4j

Graph Pattern Matching with Cypher



In Cypher

MATCH

WHERE

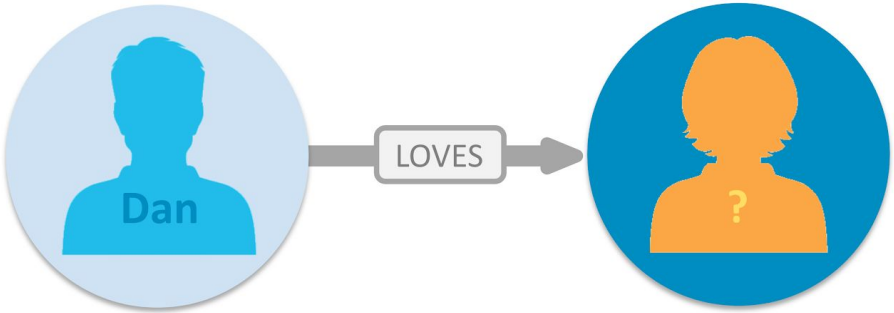
RETURN

ORDER BY

LIMIT

MindStand neo4j

Graph Pattern Matching with Cypher




```
MATCH (:Person { name:"Dan"} ) -[:LOVES]-> ( whom ) RETURN whom
```

LABEL PROPERTY VARIABLE

MindStand neo4j

Neo4j: The {Company;Product}

Product & Company started in 2007 🇸🇪

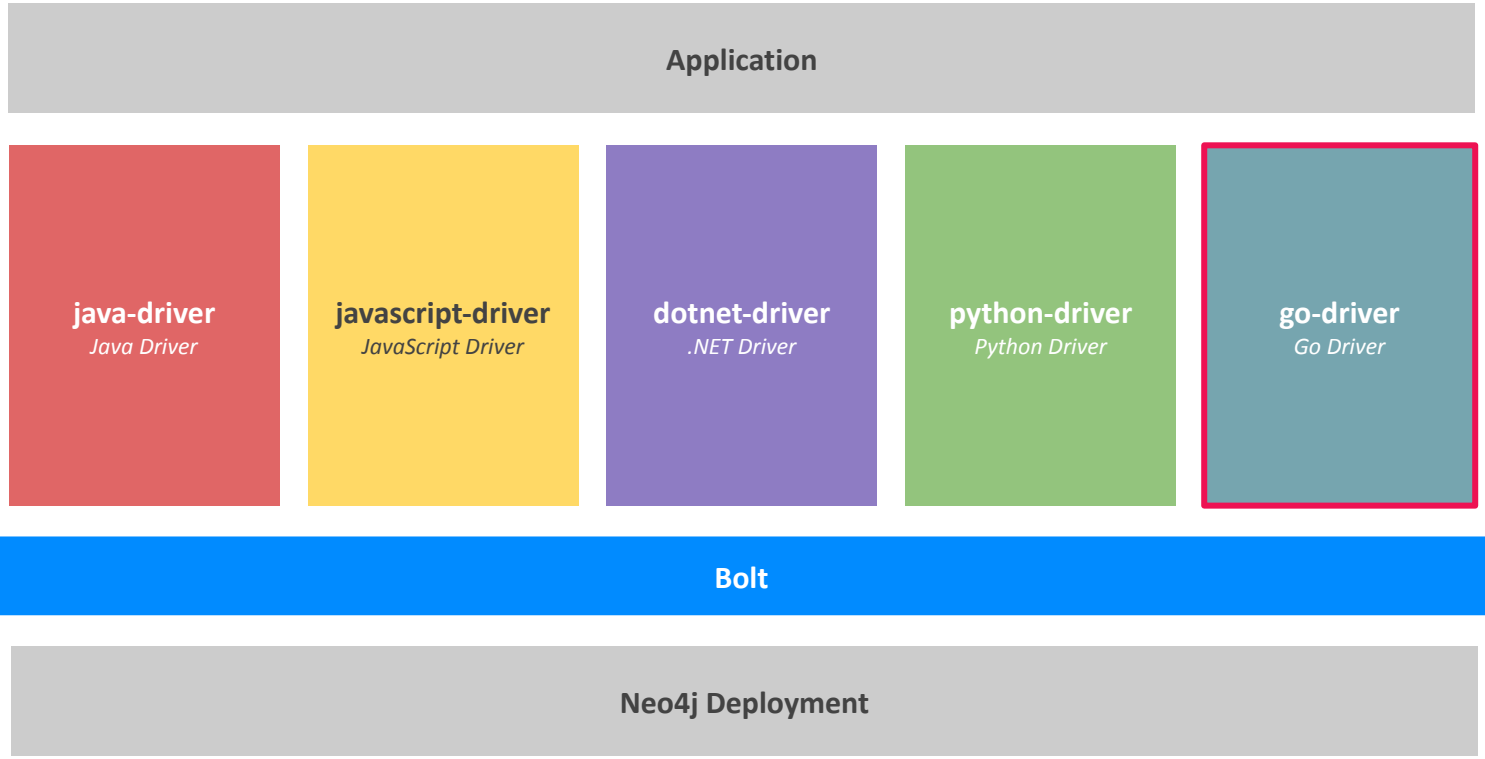
Open-Core Model

- Free Open-Source **Community** Edition
- Closed-Source **Enterprise** Edition
  - Free for local use with Neo4j Desktop / Docker

Current Version: **4.4 (LTS)** - 5.0 in the works!

Neo4j + Go

Neo4j Drivers



MindStand neo4j

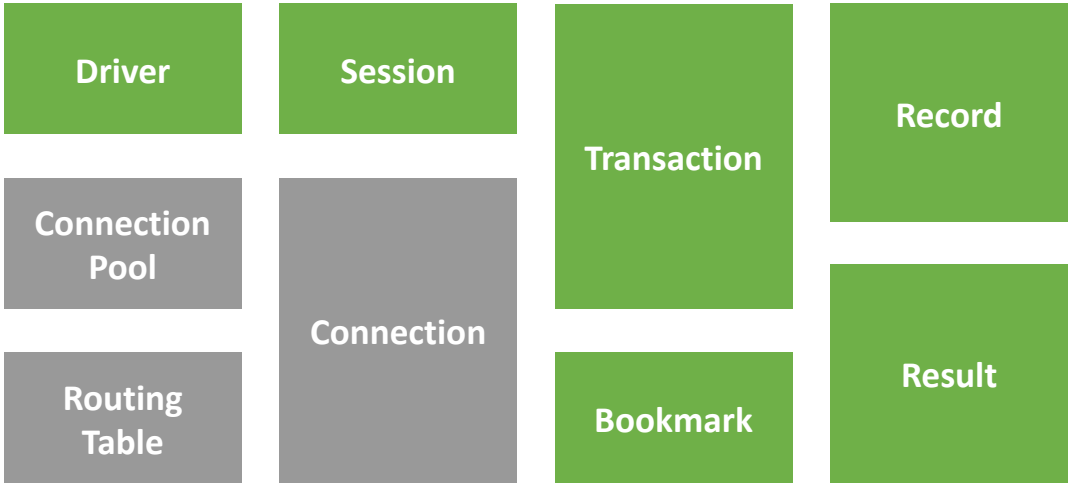
Neo4j Driver Challenges

Uniform

Idiomatic

MindStand neo4j

Neo4j Driver Concepts



MindStand neo4j

Neo4j + Go

```go
func main() {
  driver, err := neo4j.NewDriver(os.Args[1], neo4j.BasicAuth(os.Args[2], os.Args[3], ""))
  if err != nil {
     panic(err)
  }
  defer handleClose(driver)
  session := driver.NewSession(neo4j.SessionConfig{})
  defer handleClose(session)
  result, err := session.ReadTransaction(sayHello)
  if err != nil {
     panic(err)
  }
  fmt.Printf("Program says: %q", result)
}
```

Driver creation
Usually singleton
Thread-safe

Session creation
Once per thread
Sequence of causally-related transactions

Transaction execution
Retryable

MindStand neo4j

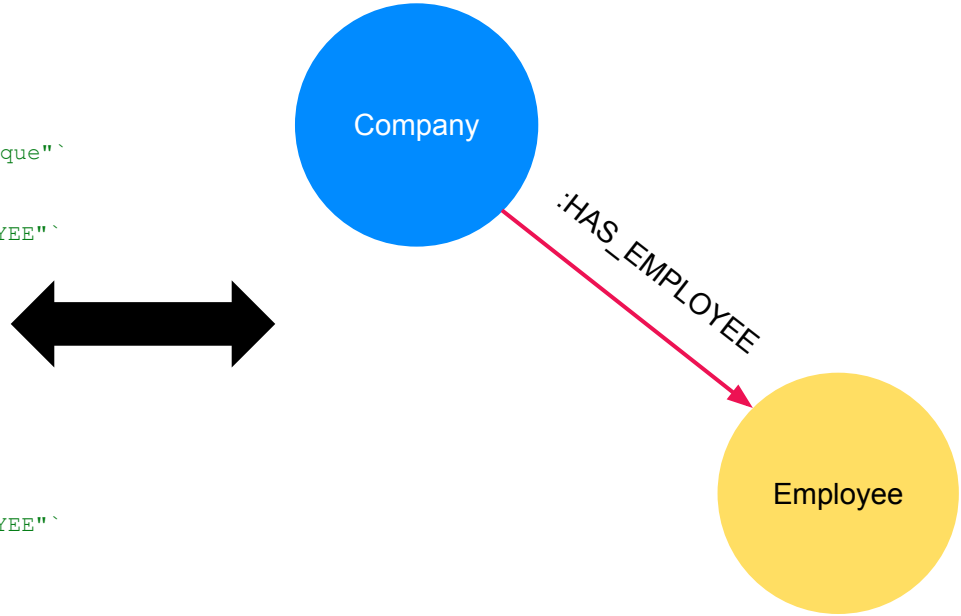
Workshop Part 2 Exercises: The Neo4j Go Driver

Neo4j + GoGM

Neo4j + GoGM - What is an OGM?

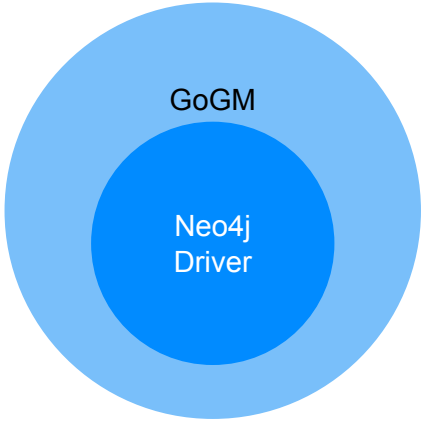
```
type Employee struct {
  gogm.BaseUUIDNode
  Name       string    `gogm:"name=name"`
  EmployeeID int       `gogm:"name=employee_id;unique"`
  Company    *Company
`gogm:"direction=incoming;relationship=HAS_EMPLOYEE"`
}

type Company struct {
  gogm.BaseUUIDNode
  Name         string      `gogm:"name=name"`
  Employees    []*Employee
`gogm:"direction=outgoing;relationship=HAS_EMPLOYEE"`
}
```



MindStand neo4j

GoGM Architecture



Neo4j Driver Responsibilities:

- (De)serialization of **primitives** *(strings, numbers, dates, etc.)*
- Communication with the database itself

GoGM Responsibilities:

- (De)serialization of **schema-defined structs** and some more advanced types like maps
- Generation of queries to transparently **save**, **update**, or **delete** a given **node and its relationships**

MindStand neo4j

Neo4j + GoGM - Defining a Schema - Node Properties

```
type ExampleNode struct {
  gogm.BaseUUIDNode
  StringField       string                 `gogm:"name=string_field"`
  UniqueInt         int                    `gogm:"name=unique_int;unique"`
  IndexedBool       bool                   `gogm:"name=indexed_bool;index"`
  GenericMap        map[string]interface{} `gogm:"name=generic_map;properties"`
  MapWithPrimitive  map[string]int         `gogm:"name=map_with_prim;properties"`
  SliceOfPrimitive  []string               `gogm:"name=prim_slice;properties"`
}
```

- GoGM utilizes Go Struct Tags to define the schema
- GoGM validates that structs are configured correctly at runtime
- Any primitive is valid as a property
- Any map of type map[string]interface{} or map[string]<primitive> is valid with a properties tag
- Any slice of type []<primitive> is also valid with a properties tag
- Fields can be marked as unique, indexed and/or primary key

MindStand neo4j

Neo4j + GoGM - Defining a Schema: Relationships

```
type LeftNode struct {
  gogm.BaseUUIDNode

  EdgeToRight       *RightNode      `gogm:"direction=incoming;relationship=EXAMPLE"`
  PropsEdgeToRight  []*ExampleEdge  `gogm:"direction=incoming;relationship=PROP_EXAMPLE"`
}

type RightNode struct {
  gogm.BaseUUIDNode

  EdgesToLeft       []*LeftNode   `gogm:"direction=outgoing;relationship=EXAMPLE"`
  PropsEdgeToRight  *ExampleEdge  `gogm:"direction=outgoing;relationship=PROP_EXAMPLE"`
}
```

- Edges must use pointers and slices of pointers
- GoGM validates that structs and relationships are configured correctly
- Both sides of a relationship must be found
- All types of directions are supported

MindStand neo4j

Neo4j + GoGM - Defining a Schema: Relationships Contd

```
// must implement gogm.Edge interface
type ExampleEdge struct {
  // would be the outgoing side
  Start *RightNode
  End   *LeftNode
  // can store same properties as a normal node
}

func (e *ExampleEdge) GetStartNode() interface{} {}
func (e *ExampleEdge) GetStartNodeType() reflect.Type {}
func (e *ExampleEdge) SetStartNode(v interface{}) error {}
func (e *ExampleEdge) GetEndNode() interface{} {}
func (e *ExampleEdge) GetEndNodeType() reflect.Type {}
func (e *ExampleEdge) SetEndNode(v interface{}) error {}
```

- Edges that store data must be defined as a struct with a Start and an End
- Edges that store data must implement the gogm.Edge interface
- For an implementation example refer to the workshop repository

Neo4j + GoGM - Using a session

```
func main() {
    g, err := gogm.New(&gogm.Config{
        // insert connection options here
    }, gogm.UUIDPrimaryKeyStrategy, &Company{}, &Employee{})
    if err != nil {
        panic(err)
    }
    // create a GoGM session
    sess, err := g.NewSessionV2(gogm.SessionConfig{})
    if err != nil {
        panic(err)
    }
    defer sess.Close() // remember to close it

    query := `MATCH p=(company:Company {name:$name})
        <-[:HAS_EMPLOYEE]-(employee:Employee)
    ) return p` // notice a path is being returned so gogm can consume it
    mindstand := Company{}
    err = sess.Query(context.Background(), query, map[string]interface{}{"name":
"MindStand"}, &mindstand)

    fmt.Printf("Company %v has %v employees", mindstand.Name, len(mindstand.Employees))
}
```

GoGM initialization
Thread-safe

Session creation
One per thread

Transaction Execution
(Non-retryable version)

MindStand neo4j

Workshop Part 3 Exercises: Using GoGM

GoGM - Coming Soon

- Near term
  - Deprecation of the gogm.Edge interface in favor of struct tags
  - Simplified Sessions
  - Diver 5.0 support
- Long term
  - Schema migration
  - Auto-generation for more advanced queries
  - GraphQL support

MindStand neo4j

Useful Links Before We Go 📸

🤓 Cypher Cheat Sheet

👩‍🎓 Neo4j GraphAcademy (including a new Go driver course!)

💻 Neo4j on Docker / on Kubernetes

Neo4j Desktop

☁️ Neo4j Sandbox

Managed Neo4j as a Service: ÅraDB (aka AuraDB)

Object-Graph Mappers: GoGM

📖 Best Practices with Neo4j Drivers

MindStand neo4j

Thank you!

GraphConnect 2022

MindStand neo4j