Data Analysis

FINAL EXAM

Made by: Juan Pablo Estrada Lucero

Primera parte

- El objetivo del código era extraer datos reales desde la API de The Movie Database (TMDB) y construir un grafo de colaboración cinematográfica del actor Laurence Fishburne.
- Cada nodo representa un actor, y cada arista muestra una colaboración en una película (con el nombre de la película como etiqueta).
- Finalmente, el programa genera una visualización gráfica en formato PNG del grafo.

Imports

```
import (
    "encoding/json"
    "fmt"
    "net/http"
    "os"
    "os/exec"
    "time"
    "github.com/awalterschulze/gographviz"
)
```

- encoding/json: para convertir respuestas JSON de la API a estructuras Go.
- net/http: para realizar solicitudes HTTP a la API de TMDB.
- os y os/exec: para crear archivos y ejecutar el comando dot de Graphviz.
- time: para manejar fechas de lanzamiento de películas.
- gographviz: para generar el grafo visual en formato DOT.

* **Formato Dot:**
  - Texto plano
  - Descripción de nodos, aristas, etc...

Estructura - Grafo

```
type Graph struct {
    nodes      map[int]string
    edges      map[[2]int]string
    degree     map[int]int
    lastMovie  map[[2]int]string
    lastDate   map[[2]int]time.Time
}
```

- nodes: almacena los actores (ID → nombre).
- edges: almacena las aristas (pares de actores → última película en común).
- degree: guarda cuántas conexiones tiene cada actor (su grado).
- lastMovie y lastDate: permiten actualizar la arista con la película más reciente si dos actores colaboraron varias veces.

Métodos - Grafo

```go
func NewGraph() *Graph {
    return &Graph{
        nodes:     make(map[int]string),
        edges:     make(map[[2]int]string),
        degree:    make(map[int]int),
        lastMovie: make(map[[2]int]string),
        lastDate:  make(map[[2]int]time.Time),
    }
}
```

- Nuevo grafo
- Inicialización de la información
- Instancia de estructura

* map = creación de mapas vácios con memoria asignada

Métodos - Grafo

```go
func (g *Graph) AddNode(id int, name string) {
    if _, exists := g.nodes[id]; !exists {
        g.nodes[id] = name
    }
}
```

- Opera sobre instancia generada
- Verificación de existencia mediante ID (evación de duplicados)
- Adición de nuevo nodo (actor)

Métodos - Grafo

```go
func (g *Graph) AddEdge(a, b int, movie string, date time.Time) {
    if a == b {
        return
    }
    key := [2]int{min(a, b), max(a, b)}

    if _, exists := g.edges[key]; !exists {
        g.edges[key] = movie
        g.lastDate[key] = date
        g.degree[a]++
        g.degree[b]++
    } else {
        if date.After(g.lastDate[key]) {
            g.edges[key] = movie
            g.lastDate[key] = date
        }
    }
}
```

- Conexiones únicas entre nodos
- Creación de aristas
- Almacenamiento de película y fecha de colaboración
- Evita conexión entre el mismo nodo
- Actualización de los datos de la película más reciente

Métodos - Grafo

```go
func (g *Graph) TotalNodes() int {
    return len(g.nodes)
}
```

- Número total de nodos (actores) almacenados.
- Clave = ID actor Valor = Nombre de actor
- Indica cantidad de actores en el grafo

Métodos - Grafo

```go
func (g *Graph) TotalEdges() int {
    return len(g.edges)
}
```

- Número total de aristas entre nodos del grafo.

- Clave = Par de ID’s de actores
- Valor = Nombre de película de colaboración

Métodos - Grafo

```go
func (g *Graph) MaxDegreeNodes() map[int]string {
    maxDeg := 0
    for _, d := range g.degree {
        if d > maxDeg {
            maxDeg = d
        }
    }
    result := make(map[int]string)
    for id, d := range g.degree {
        if d == maxDeg {
            result[id] = g.nodes[id]
        }
    }
    return result
}
```

- Nodo más conectado (centríco de la red)
- Almacena valor de las conexiones de un nodo (grado)

Métodos - Grafo

```go
func min(a, b int) int {
    if a < b {
        return a
    }
    return b
}

func max(a, b int) int {
    if a > b {
        return a
    }
    return b
}
```

- Normaliza pares de actores que forman una conexión (relación)
- Permite identificar claves diferentes en los mapas de Go.
- Empleado en AddAge

TMDBAPIUtils

```go
type TMDBAPIUtils struct {
    APIKey string
}

type CastMember struct {
    ID    int    `json:"id"`
    Name  string `json:"name"`
    Order int    `json:"order"`
}

type MovieCredit struct {
    ID          int    `json:"id"`
    Title       string `json:"title"`
    ReleaseDate string `json:"release_date"`
}
```

TMDBAPIUtils:

- Objeto de utilidad para interactuar con el API de TMDB. (Estructura)
- Receptor de los métodos para realizar llamados.

CastMember:

- Información de actor
- Mapeo de datos

MovieCredit:

- Información de película
- Indica que películas conectan a los actores dentro del grafo

TMDBUtils

```go
func (api *TMDBAPIUtils) GetMovieCast(movieID string, limit int, excludeIDs []int) ([]CastMember, error) {
    url := fmt.Sprintf("%s/movie/%s/credits?api_key=%s", baseURL, movieID, api.APIKey)
    resp, err := http.Get(url)
    if err != nil {
        return nil, err
    }
    defer resp.Body.Close()

    var result struct {
        Cast []CastMember `json:"cast"`
    }
    if err := json.NewDecoder(resp.Body).Decode(&result); err != nil {
        return nil, err
    }

    exclude := make(map[int]bool)
    for _, id := range excludeIDs {
        exclude[id] = true
    }

    finalCast := []CastMember{}
    for _, member := range result.Cast {
        if exclude[member.ID] {
            continue
        }
        if len(finalCast) < limit {
            finalCast = append(finalCast, member)
        }
    }
    return finalCast, nil
}
```

GetMovieCast(movieID, limit, excludeIDs):

- Consulta los actores que participaron en una película específica.
- Aplica un límite (limit) y puede excluir actores específicos (por ejemplo, el actor principal para no duplicarlo).
- Devuelve una lista de CastMember (id, nombre, orden).

TMDBUtils

```go
func (api *TMDBAPIUtils) GetMovieCreditsForPerson(personID, startDate, endDate string) ([]MovieCredit, error) {
    url := fmt.Sprintf("%s/person/%s/movie_credits?api_key=%s", baseURL, personID, api.APIKey)
    resp, err := http.Get(url)
    if err != nil {
        return nil, err
    }
    defer resp.Body.Close()

    var result struct {
        Cast []MovieCredit `json:"cast"`
    }
    if err := json.NewDecoder(resp.Body).Decode(&result); err != nil {
        return nil, err
    }

    layout := "2006-01-02"
    start, _ := time.Parse(layout, startDate)
    end, _ := time.Parse(layout, endDate)

    filtered := []MovieCredit{}
    for _, movie := range result.Cast {
        date, err := time.Parse(layout, movie.ReleaseDate)
        if err == nil && date.After(start) && date.Before(end) {
            filtered = append(filtered, movie)
        }
    }
    return filtered, nil
}
```

GetMovieCreditsForPerson(personID, startDate, endDate):

- Devuelve todas las películas en las que actuó una persona.
- Filtra las películas por rango de fechas (por ejemplo, desde 2000 hasta 2025).
- Devuelve una lista de películas con su título y fecha de lanzamiento.

Generación del grafo

```go
func GenerateGraphViz(g *Graph) {
    graphAst, _ := gographviz.ParseString(`digraph G {}`)
    graphObj := gographviz.NewGraph()
    gographviz.Analyse(graphAst, graphObj)

    graphObj.SetDir(false)
    graphObj.AddAttr("G", "splines", "true")
    graphObj.AddAttr("G", "overlap", "false")

    for id, name := range g.nodes {
        graphObj.AddNode("G", fmt.Sprintf("n%d", id), map[string]string{
            "label": fmt.Sprintf("\"%s\"", name),
            "shape": "ellipse",
            "style": "filled",
            "fillcolor": "\"■#cce5ff\"",
        })
    }

    for edge, movie := range g.edges {
        src := fmt.Sprintf("n%d", edge[0])
        dst := fmt.Sprintf("n%d", edge[1])
        graphObj.AddEdge(src, dst, false, map[string]string{
            "label": fmt.Sprintf("\"%s\"", movie),
        })
    }

    dotOutput := "graph.dot"
    pngOutput := "graph.png"
    os.WriteFile(dotOutput, []byte(graphObj.String()), 0644)

    cmd := exec.Command("dot", "-Tpng", dotOutput, "-o", pngOutput)
    err := cmd.Run()
    if err != nil {
        fmt.Println("⚠️ Could not generate PNG:", err)
        return
    }
    fmt.Printf("\n✅ Graph visualization saved as %s\n", pngOutput)
}
```

- Esta función convierte la información del grafo en una imagen .png.
- Se crea un grafo con gographviz.
- Se agregan nodos (actores) con color y estilo
- Se agregan aristas con el nombre de la película como etiqueta
- Se genera un archivo .dot y luego se convierte a .png usando Graphviz

Segunda parte

- El objetivo de esta parte consistía en cargar los datos en formato .csv a una base de datos relacional, para posteriormente tener un acercamiento con la creación de índices y tablas, para mostrar los datos a través de una herramienta de visualización.

Carga de datos

- 1. Creación de base de datos y usuario:

```
CREATE DATABASE incidents_db CHARACTER SET = 'utf8mb4' COLLATE =
'utf8mb4_unicode_ci';

CREATE USER 'incuser'@'localhost' IDENTIFIED BY 'IncPass123!';
GRANT ALL PRIVILEGES ON incidents_db.* TO 'incuser'@'localhost';
FLUSH PRIVILEGES;
```

Carga de datos

- 2. Creación de tablas:

```
USE incidents_db;

CREATE TABLE incidents (
report_id VARCHAR(128) NOT NULL PRIMARY
KEY,
category VARCHAR(255),
date DATE
);

CREATE TABLE details (
report_id VARCHAR(128) NOT NULL PRIMARY
KEY,
subject VARCHAR(512),
transport_mode VARCHAR(255),
detection VARCHAR(255),
FOREIGN KEY (report_id) REFERENCES
incidents(report_id) ON DELETE CASCADE
);

CREATE TABLE outcomes (
report_id VARCHAR(128) NOT NULL PRIMARY
KEY,
outcome VARCHAR(255),
num_ppl_fined INT,
fine DOUBLE,
num_ppl_arrested INT,
prison_time DOUBLE,
prison_time_unit VARCHAR(64),
FOREIGN KEY (report_id) REFERENCES
incidents(report_id) ON DELETE CASCADE
);
```

Carga de datos

- 3. Creación de índices:

```
CREATE INDEX idx_incidents_report_id ON incidents (report_id);

CREATE INDEX idx_details_report_id ON details (report_id);

CREATE INDEX idx_outcomes_report_id ON outcomes (report_id);
```

Carga de datos

- 4.Importar datos mediante LOAD DATA INFILE:

- **Edición de archivo de configuración de mysql:**

[mysqld]
local_infile=1

- **Comandos para carga de datos:**

```
-- IMPORT outcomes.csv
LOAD DATA LOCAL INFILE '/home/sir_anon/csvs/outcomes.csv'
INTO TABLE outcomes
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(report_id, outcome, num_ppl_fined, fine, num_ppl_arrested,
prison_time, prison_time_unit);
```

```
-- IMPORT outcomes.csv
LOAD DATA LOCAL INFILE '/home/sir_anon/csvs/
outcomes.csv'
INTO TABLE outcomes
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(report_id, outcome, num_ppl_fined, fine,
num_ppl_arrested, prison_time, prison_time_unit);
```

```
-- IMPORT details.csv
LOAD DATA LOCAL INFILE '/home/sir_anon/csvs/details.csv'
INTO TABLE details
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(report_id, subject, transport_mode, detection);unit);
```

Visualización de datos

- 1. Instalación de metabase:

- sudo apt install openjdk-17-jre
- wget https://downloads.metabase.com/v0.48.6/metabase.jar -O metabase.jar
- java -jar metabase.jar

- 2. Vinculación de metabase con base de datos:

Para ello se llevó a cabo el uso de la GUI proporcionada por metabase en la dirección web para abrirlo de forma local

(a) Porcentaje de incidentes entre 2018-01-01 y 2020-12-31

Query:

```sql
SELECT
  ROUND(
    (SUM(CASE WHEN date BETWEEN '2018-01-01' AND '2020-12-31' THEN 1 ELSE 0 END) / COUNT(*)) * 100,
    2
  ) AS Porcentaje_de_incidentes_entre_2018_y_2020
FROM incidents;
```

- CASE WHEN: Revisa cada fila de la tabla incidents. (+1 o +0)
- Cálculo del porcentaje

(b) Tres métodos de transporte más comunes detectados por “intelligence”

Query:

```sql
SELECT
  transport_mode as Transport_mode,
  COUNT(*) AS Total
FROM details
WHERE detection LIKE '%intelligence%'
  AND transport_mode IS NOT NULL
  AND transport_mode != ''
GROUP BY transport_mode
ORDER BY total DESC
LIMIT 3;
```

- WHERE detection LIKE '%intelligence%': Filtra registros donde el método de detección contenga la plabra intelligence

(c) Métodos de detección con el mayor promedio de gente arrestada

Query:

```sql
SELECT
  CASE
    WHEN LOWER(d.detection) LIKE '%intelligence%' THEN 'Intelligence'
    WHEN LOWER(d.detection) LIKE '%inspection%' THEN 'Routine Inspection'
    WHEN LOWER(d.detection) LIKE '%x-ray%' THEN 'X-ray'
    WHEN LOWER(d.detection) LIKE '%operation%' THEN 'Operation'
    WHEN LOWER(d.detection) LIKE '%investigation%' THEN 'Investigation'
    WHEN LOWER(d.detection) LIKE '%target%' THEN 'Targeting'
    WHEN LOWER(d.detection) LIKE '%risk%' THEN 'Risk Assessment'
    WHEN LOWER(d.detection) LIKE '%test%' THEN 'Test Purchase'
    WHEN LOWER(d.detection) LIKE '%dog%' THEN 'Dogs'
    WHEN LOWER(d.detection) LIKE '%online%' THEN 'Online'
    WHEN LOWER(d.detection) LIKE '%other%' THEN 'Other'
    WHEN LOWER(d.detection) LIKE '%drone%' THEN 'Drone'
    WHEN LOWER(d.detection) LIKE '%lake%' OR LOWER(d.detection) LIKE '%river%' THEN 'Lake/River'
    WHEN LOWER(d.detection) LIKE '%train%' THEN 'Land - Train'
    WHEN LOWER(d.detection) LIKE '%foot%' THEN 'Land - Foot'
    WHEN LOWER(d.detection) LIKE '%vehicle%' THEN 'Land - Vehicle'
    WHEN LOWER(d.detection) LIKE '%air%' THEN 'Air'
    WHEN LOWER(d.detection) LIKE '%sea%' THEN 'Sea'
    ELSE 'Other/Unclassified'
  END AS metodo_deteccion,
  ROUND(AVG(o.num_ppl_arrested), 2) AS promedio_arrestados
FROM details d
JOIN outcomes o ON d.report_id = o.report_id
WHERE d.detection IS NOT NULL
  AND d.detection != ''
GROUP BY metodo_deteccion
ORDER BY promedio_arrestados DESC;
```

- CASE WHEN: Clasifica los métodos de detección (por ejemplo, “x-ray”, “dog”, “operation”) en categorías estandarizadas, unificando nombres similares.

- JOIN entre details y outcomes: Relaciona los datos descriptivos del caso (details) con los resultados (outcomes) mediante el report_id

(d) Categorías con las sentencias de prisión más largas

Query:

```sql
SELECT
  i.category,
  ROUND(AVG(
    CASE
      WHEN LOWER(TRIM(o.prison_time_unit)) LIKE 'year%'
           AND TRIM(o.prison_time) REGEXP '^[0-9]+(\\.[0-9]+)?$'
        THEN CAST(TRIM(o.prison_time) AS DECIMAL(10,2)) * 365
      WHEN LOWER(TRIM(o.prison_time_unit)) LIKE 'month%'
           AND TRIM(o.prison_time) REGEXP '^[0-9]+(\\.[0-9]+)?$'
        THEN CAST(TRIM(o.prison_time) AS DECIMAL(10,2)) * 30
      WHEN LOWER(TRIM(o.prison_time_unit)) LIKE 'day%'
           AND TRIM(o.prison_time) REGEXP '^[0-9]+(\\.[0-9]+)?$'
        THEN CAST(TRIM(o.prison_time) AS DECIMAL(10,2))
      ELSE NULL
    END
  ), 2) AS promedio_dias_prision,

  ROUND(SUM(
    CASE
      WHEN LOWER(TRIM(o.prison_time_unit)) LIKE 'year%'
           AND TRIM(o.prison_time) REGEXP '^[0-9]+(\\.[0-9]+)?$'
        THEN CAST(TRIM(o.prison_time) AS DECIMAL(10,2)) * 365
      WHEN LOWER(TRIM(o.prison_time_unit)) LIKE 'month%'
           AND TRIM(o.prison_time) REGEXP '^[0-9]+(\\.[0-9]+)?$'
        THEN CAST(TRIM(o.prison_time) AS DECIMAL(10,2)) * 30
      WHEN LOWER(TRIM(o.prison_time_unit)) LIKE 'day%'
           AND TRIM(o.prison_time) REGEXP '^[0-9]+(\\.[0-9]+)?$'
        THEN CAST(TRIM(o.prison_time) AS DECIMAL(10,2))
      ELSE NULL
    END
  ), 2) AS total_dias_prision
FROM incidents i
JOIN outcomes o
  ON i.report_id = o.report_id
WHERE o.prison_time IS NOT NULL
  AND TRIM(o.prison_time) <> ''
GROUP BY i.category
ORDER BY total_dias_prision DESC;
```

- Conversión de las fechas de años o meses a días.
- Agrupación por categoría
- Orden de mayor a menor

(e) Serie de tiempo anual con totales de multa por año

Query:

```sql
SELECT
  YEAR(i.date) AS anio,
  SUM(o.fine) AS Total_multas
FROM incidents i
JOIN outcomes o ON i.report_id = o.report_id
GROUP BY anio
ORDER BY anio;
```

- JOIN: Relaciona información de incidentes con la información de resultados

FIN