

Carlos Ariel Rubio Espinosa

Creación de Schema:

```
<country>: string @index(term) .  
<dgraph.drop.op>: string .  
<dgraph.graphql.p_query>: string @index(sha256) .  
<dgraph.graphql.schema>: string .  
<dgraph.graphql.xid>: string @index(exact) @upsert .  
<dgraph.namespace.id>: int @index(int) @upsert .  
<dgraph.namespace.name>: string @index(exact) @upsert .  
<duration>: int .  
<genre>: string @index(hash) .  
<has_song>: uid @reverse .  
<label>: string @index(hash) .  
<name>: string @index(exact, fulltext) .  
<part_of>: uid @reverse .  
<release_year>: int @index(int) .  
<sung_by>: uid @reverse .  
<title>: string .  
type <dgraph.graphql> {  
    dgraph.graphql.schema  
    dgraph.graphql.xid  
}  
type <dgraph.graphql.persisted_query> {  
    dgraph.graphql.p_query  
}  
type <dgraph.namespace> {  
    dgraph.namespace.name
```

dgraph.namespace.id

}

Prueba del Schema hecho:

The screenshot shows the dGraph schema editor interface. On the left is a sidebar with icons for Console, Schema, Cluster, Backups, and Help. The main area has tabs for Schema, Add Predicate, Predicates, Types, Bulk Edit, Refresh Schema, and Properties. The Schema tab is active, displaying a table of predicates:

Predicate	Type	Indices
country	string	term
dgraph.drop.op	string	
dgraph.graphql.p_query	string	sha256
dgraph.graphql.schema	string	
dgraph.graphql.xid	string	exact
dgraph.namespace.id	int	int
dgraph.namespace.name	string	exact
duration	int	
genre	string	hash
has_song	uid	
label	string	hash
name	string	exact, fulltext
part_of	uid	
release_year	int	int
sung_by	uid	
title	string	

Inserción de datos:

The screenshot shows the dGraph console interface. On the left is a sidebar with icons for Console, Schema, Cluster, Backups, and Help. The main area has tabs for Query, Mutate, Clear, Run, Message, JSON, Request, Geo, and HISTORY. The Mutate tab is active, showing a JSON mutation query:

```
gset { "set": [ { "uid": "...", "song1": { "title": "Bohemian Rhapsody", "release_year": 1975, "duration": 5.55, "sung_by": { ... }, "part_of": { ... } }, "uid": "...", "song2": { "title": "Hey Jude", "release_year": 1968, "duration": 7.11, "sung_by": { "uid": "...", "artist2": { "part_of": { "uid": "...", "album2": { "uid": "...", "artist1": { "name": "Queen", "country": "UK", "genre": "Rock", "has_song": [ { "uid": "...", "song1": { ... } } ] }, "artist2": { "name": "The Beatles", "country": "UK", "genre": "Rock", "has_song": [ { "uid": "...", "song2": { ... } } ] } } } } } ] }
```

The message area shows "Message: Done". Below the message are two history entries:

```
gset { "set": [ { "uid": "...", "song1": { "title": "Bohemian Rhapsody", "release_year": 1975, "duration": 5.55, "sung_by": { ... }, "part_of": { ... } }, "uid": "...", "song2": { "title": "Hey Jude", "release_year": 1968, "duration": 7.11, "sung_by": { "uid": "...", "artist2": { "part_of": { "uid": "...", "album2": { "uid": "...", "artist1": { "name": "Queen", "country": "UK", "genre": "Rock", "has_song": [ { "uid": "...", "song1": { ... } } ] }, "artist2": { "name": "The Beatles", "country": "UK", "genre": "Rock", "has_song": [ { "uid": "...", "song2": { ... } } ] } } } } } ] }
```

```
gset { "set": [ { "uid": "...", "song1": { "title": "Bohemian Rhapsody", "release_year": 1975, "duration": 5.55, "sung_by": { ... }, "part_of": { ... } }, "uid": "...", "song2": { "title": "Hey Jude", "release_year": 1968, "duration": 7.11, "sung_by": { "uid": "...", "artist2": { "part_of": { "uid": "...", "album2": { "uid": "...", "artist1": { "name": "Queen", "country": "UK", "genre": "Rock", "has_song": [ { "uid": "...", "song1": { ... } } ] }, "artist2": { "name": "The Beatles", "country": "UK", "genre": "Rock", "has_song": [ { "uid": "...", "song2": { ... } } ] } } } } } ] }
```

Primera Consulta:

localhost:8080

Console

```

1 {
2   artist(func: eq(name, "Queen")) {
3     uid
4     name
5     country
6     has_song {
7       uid
8       title
9       release_year
10      duration
11    }
12  }
13

```

Graph JSON Request Geo

Q { artist(func: eq(name, "Queen")) { uid name country has_song { ... } } }

uid: 0x5

pred.	value
duration	5
release_year	1975
title	'Bohemian Rhapsody'
uid	'0x5'

Segunda Consulta:

localhost:8080

Console

```

1 {
2   song(func: eq(title, "Hey Jude")) {
3     uid
4     title
5     # Sigue la relación "part_of" para ver el álbum
6     part_of {
7       uid
8       title
9       release_year
10      label
11    }
12  }
13

```

Graph JSON Request Geo

Q # ESTA CONSULTA SOLO FUNCIONA SI AÑADES @index(exact) A <title> { song(func: eq(title, "Hey Jude")) { uid title # Sigue la relación "part_of" para ... } }

Showing 2 nodes and 1 edges

Tercera Consulta:

The screenshot shows the Dgraph console interface. On the left, the sidebar includes 'Console', 'Schema', 'Cluster', and 'Backups'. The main area has tabs for 'Query' (selected), 'Mutate', and 'Clear'. A query is entered in the text input:

```

1 {
2   songs_by_year(func: eq(release_year, 1975))
3     @filter(has(duration)) {
4       uid
5       title
6       release_year
7       duration
8       sung_by {
9         name
10      }
11    }

```

The results pane shows a graph with a green node labeled 'Queen' connected to a blue node labeled 'Bohemian Rhapsody' via a green edge labeled 'sung_by'. Below the graph, a table provides details for the song:

uid: 0x5	
pred.	value
duration	5
release_year	1975
title	'Bohemian Rhapsody'
uid	'0x5'

Mostrar todos los datos:

The screenshot shows the Dgraph console interface. The sidebar includes 'Console', 'Schema', 'Cluster', and 'Backups'. The main area has tabs for 'Query' (selected), 'Mutate', and 'Clear'. A query is entered in the text input:

```

1 {
2   artista(func: has(name)){
3     uid
4     name
5     has_song{
6       uid
7       title
8       part_of{
9         uid
10      title
11      release_year
12    }
13  }
14 }
15

```

The results pane shows a graph with nodes for 'The Beatles' (blue) and 'Queen' (green). They are connected by a green edge labeled 'has_song'. Each node also has a pink 'part_of' edge pointing to a green node labeled 'The Beatles' and 'Queen' respectively. The status bar at the bottom right indicates 'Showing 6 nodes and 4 edges'.

Conclusión:

Al ser esto una creación y consulta más guiada, siento que es más fácil de entender cómo funciona Dgraph y más rápida de hacer esto que el primer laboratorio.