Neo4j Database Management System

Akshay Pudage
Kushal Gevaria

Problems with other DBMS

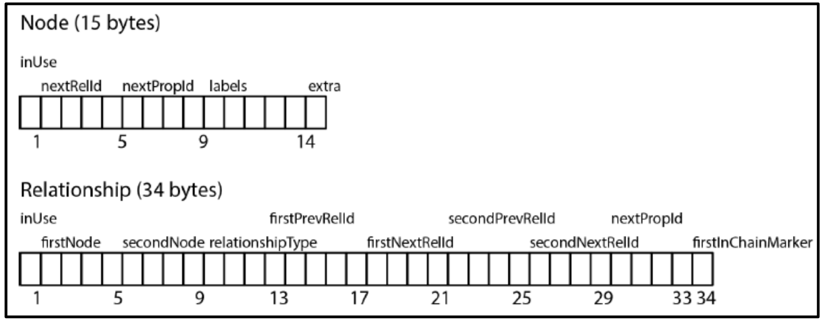
- Traditional relational DBMS cannot handle unstructured data and ad-hoc relations
- Relations in RDBMS need expensive join operations
- NoSQL databases are bad at handling relations
- Reciprocal queries can be very expensive

Why Neo4j?

- Uses graph data structure for modeling data
- Relations can be easily represented in graphs
- Graph traversals take O(1) time

Data Storage & Indexing

- Uses index-free adjacency which speeds up traversals
- Native graph processing capability
- Maintain separate stores for storing nodes, relationships and properties.
- Uses fixed record size to enable O(1) lookups
- Forms a linked list internally for navigating across records



Query Processing and Optimization

- Cypher: The most declarative language
- CRUD operations like create, read, update and delete operations can be done on the database using simple Cypher queries
- Cypher query language represents all the complex relations and structures in a meaningful and concise way
- Cypher provides a special operator to find the neighborhood nodes the way we want. In traditional relational database, this is cumbersome with multiple joins and cartesian products.

Query Processing and Optimization

- Regular search over the entire data can be costly in terms of time and resources
- Cypher provides an option of indexing. We can create an index on a single attribute or composite indexes are also available
- Composite indexes allow index creation on more than one attribute
- The Neo4j tries to execute the queries as fast as possible and in order to do that sometimes a query tuning
- Each cypher query gets optimized and transformed into an execution plan.
- Execution plan: Uses minimal resources chosen from this set of plans.
- For parameterized queries as oppose to hardcoded literals the query engine also re-uses the same execution plan instead of building all plans and choosing the best.
- Furthermore, there are several operators that aid in choosing the best query plan.

Transaction Management & Security

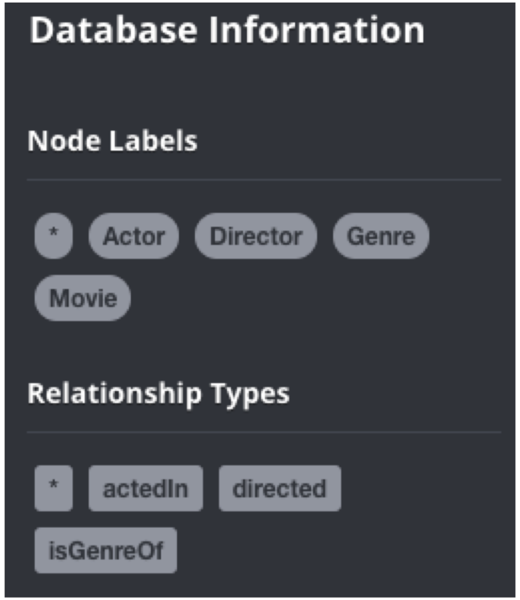
- Uses two-commit transactions
- Lock manager acquires write locks on nodes and relationships
- If successful, changes are flushed to disk and locks are released
- If failure, changes are discarded and locks are released
- Creates a transaction object internally which keeps track of changes
- Uses Write Ahead Log while committing changes to disk

Neo4j NoSQL Database Application

- Neo4j community version 3.2.5: Application UI that operates conveniently across all the platforms
- Backend is supported by Java: Neo4j works best with Java and is supported by all the packages in Java
- “Org.neo4j.graphdb”: To establish the connection with the database, using "GraphDatabaseService" class
- “GraphDatabaseService" package provides an instance of Neo4j graphDB to perform CRUD operations on the nodes and relationships in java

Database Information

- Nodes are created with their respective labels
- Movie, Actor, Director, Genre are the type of nodes. Here, type is displayed by the labels assigned to those nodes
- Relations are created between nodes. Each relation has a relation type. It can be directed or undirected.
- For IMDB dataset, we created three basic relations between nodes.
- “ActedIn”: Relation between an actor and a movie
- “Directed”: Relation between a director and a movie
- “IsGenreOf”: Relation between a genre and a movie

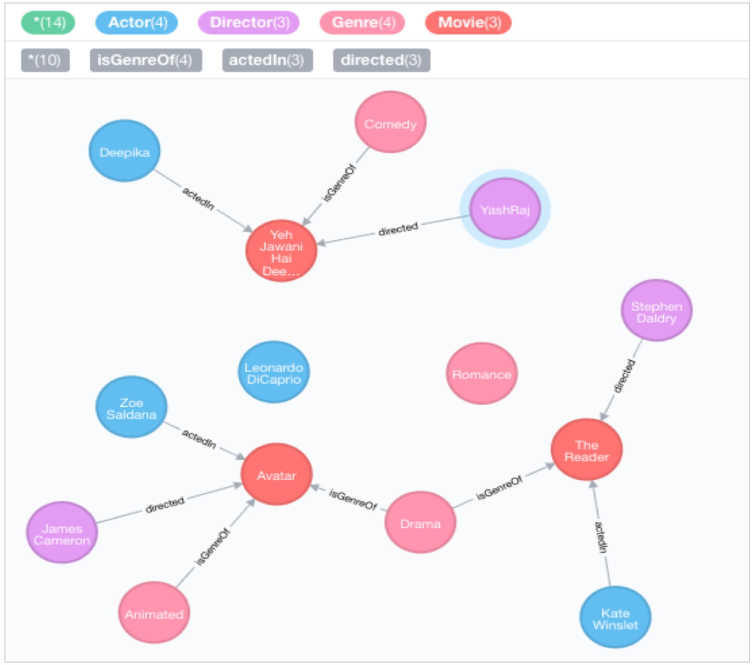

Database Information

Node Labels

Relationship Types

Neo4j Sample Graph for IMDB

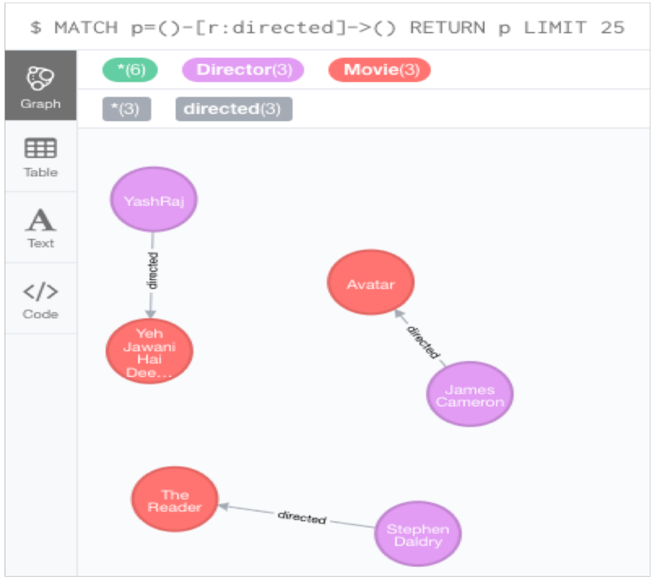
- Small subgraph of the IMDB dataset after certain CRUD operations is displayed on the right
- There are currently 4 actors, 3 directors, 4 genres and 3 movies in this sample graph.
- Whole graph has millions of nodes with complex relationships between multiple nodes.
- Movie “Avatar” has two genres namely “Animated”, “Drama”, one director “James Cameron” and an actress “Zoe Saldana”. (In reality there is more to this graph)



Relationship between Directors and Movies

- On the right is the example of Cypher query to get all the relations between directors and movies
- Similar can be achieved in Java ->

```
Node dNode = getDbsiIMDB().findNode(directorLabel,
"name", directorName);
for(Relationship r : dNode.getRelationships()) {
        // get nodes on other end of this relationship
        Node tempMovie = r.getEndNode();
}
```

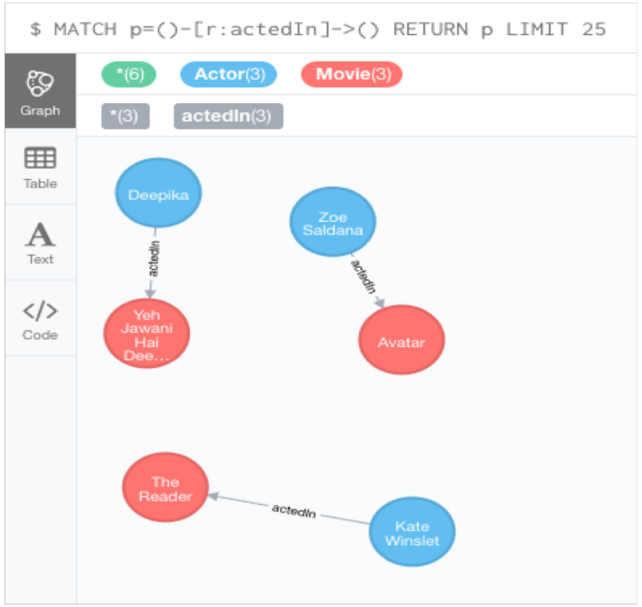


```
$ MATCH p=()-[r:directed]->() RETURN p LIMIT 25
```

Relationship between Actors and Movies

- On the right is the example of Cypher query to get all the relations between actors and movies
- Similar can be achieved in Java ->

```
Node aNode = getDbsiIMDB().findNode(actorLabel,
"name", actorName);
for(Relationship r : aNode.getRelationships()) {
                // get nodes on other end of this relationship
                Node tempMovie = r.getEndNode();
}
```

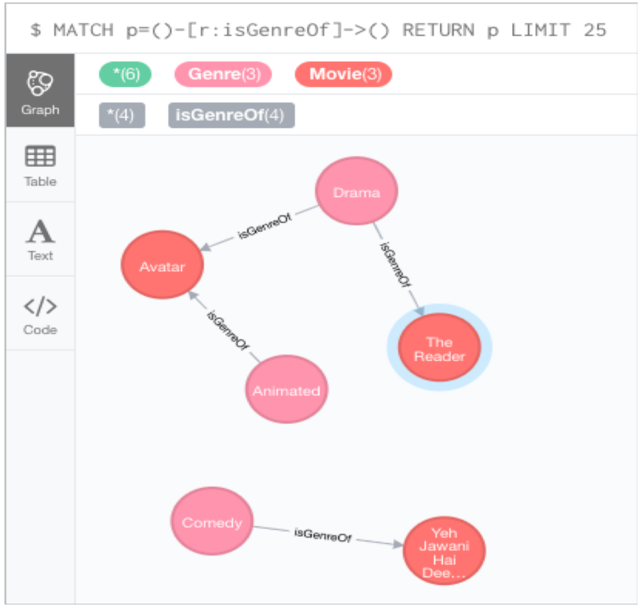


```
$ MATCH p=()-[r:actedIn]->() RETURN p LIMIT 25
```

Relationship between Genres and Movies

- On the right is the example of Cypher query to get all the relations between genres and movies
- Similar can be achieved in Java ->

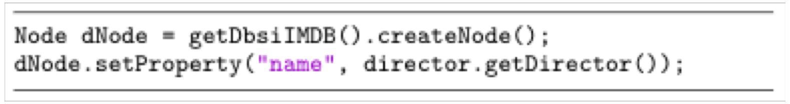
```
Node gNode = getDbsiIMDB().findNode(genreLabel,
"name", genreName);
for(Relationship r : gNode.getRelationships()) {
        // get nodes on other end of this relationship
        Node tempMovie = r.getEndNode();
}
```



```
$ MATCH p=()-[r:isGenreOf]->() RETURN p LIMIT 25
```

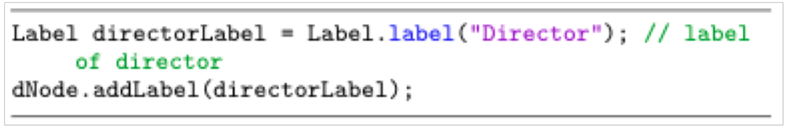
Create Nodes and Relationships using Java

Example to create a director node:



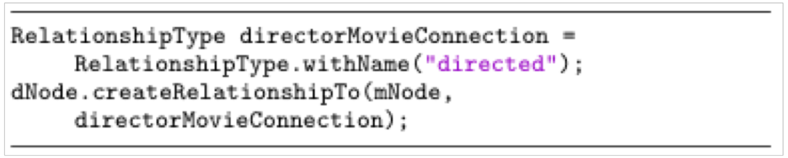
```
Node dNode = getDbsiIMDB().createNode();
dNode.setProperty("name", director.getDirector());
```

Example to create a label for all director nodes:



```
Label directorLabel = Label.label("Director"); // label
     of director
dNode.addLabel(directorLabel);
```

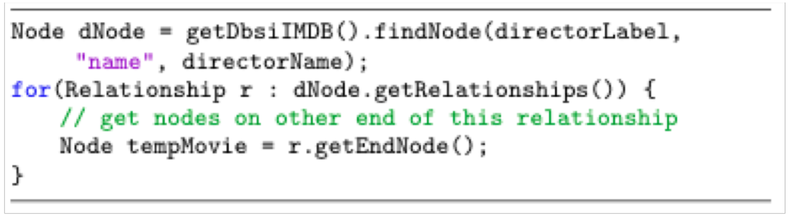
Example to create a relationship type for all director nodes:



```
RelationshipType directorMovieConnection =
     RelationshipType.withName("directed");
dNode.createRelationshipTo(mNode,
     directorMovieConnection);
```

Find Nodes and Relationships using Java

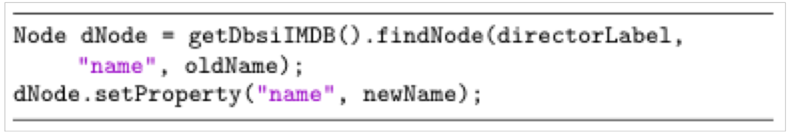
Example to find a director node and its relationships::



```java
Node dNode = getDbsiIMDB().findNode(directorLabel,
      "name", directorName);
for(Relationship r : dNode.getRelationships()) {
    // get nodes on other end of this relationship
    Node tempMovie = r.getEndNode();
}
```

Update Nodes using Java

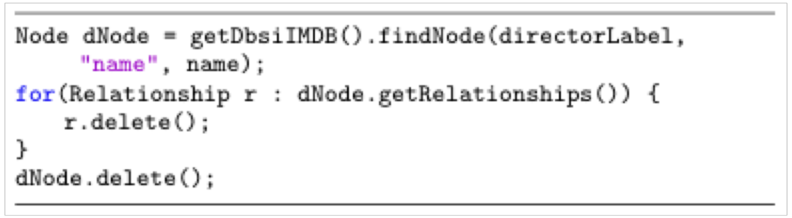
Example to update a director node:



```
Node dNode = getDbsiIMDB().findNode(directorLabel,
     "name", oldName);
dNode.setProperty("name", newName);
```

Delete Nodes and its Relationships using Java

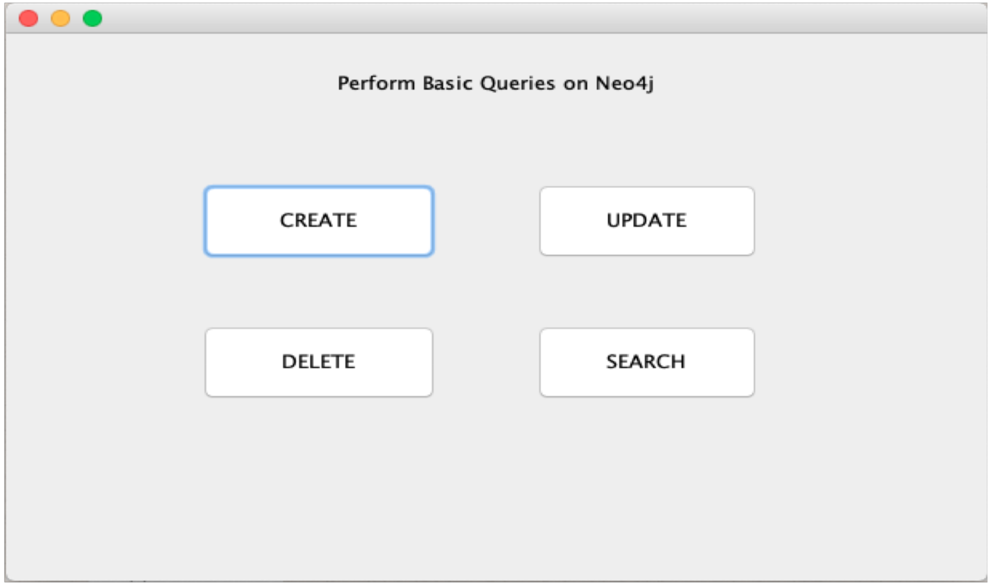
Example to delete a director node and its relationships:



```
Node dNode = getDbsiIMDB().findNode(directorLabel,
      "name", name);
for(Relationship r : dNode.getRelationships()) {
    r.delete();
}
dNode.delete();
```

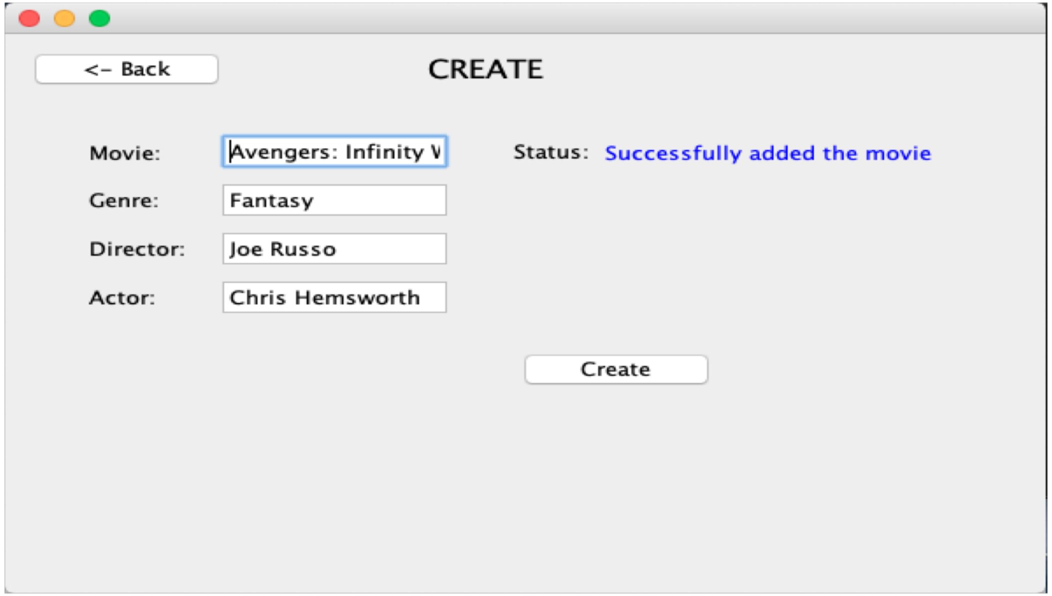
Neo4j GUI Application using Java Spring Builder

CRUD Operation Panel



Neo4j GUI Application: CREATE MOVIE

Create a movie

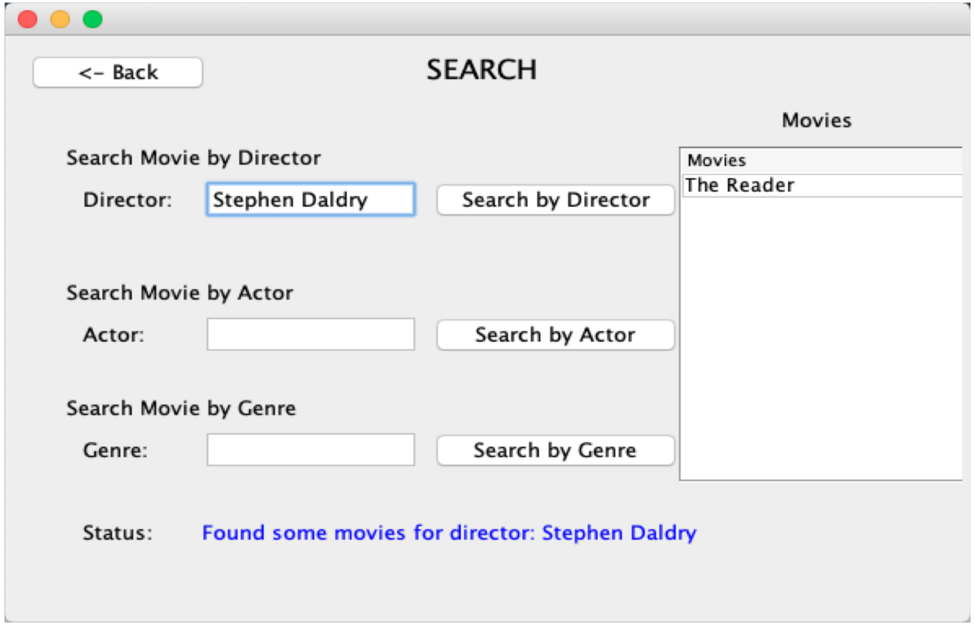


Movie:	Avengers: Infinity V
Genre:	Fantasy
Director:	Joe Russo
Actor:	Chris Hemsworth

Status: Successfully added the movie

Neo4j GUI Application: SEARCH BY DIRECTOR

Search Movies by Director



<- Back

SEARCH

Search Movie by Director

Director: Stephen Daldry

Search by Director

Search Movie by Actor

Actor:

Search by Actor

Search Movie by Genre

Genre:

Search by Genre

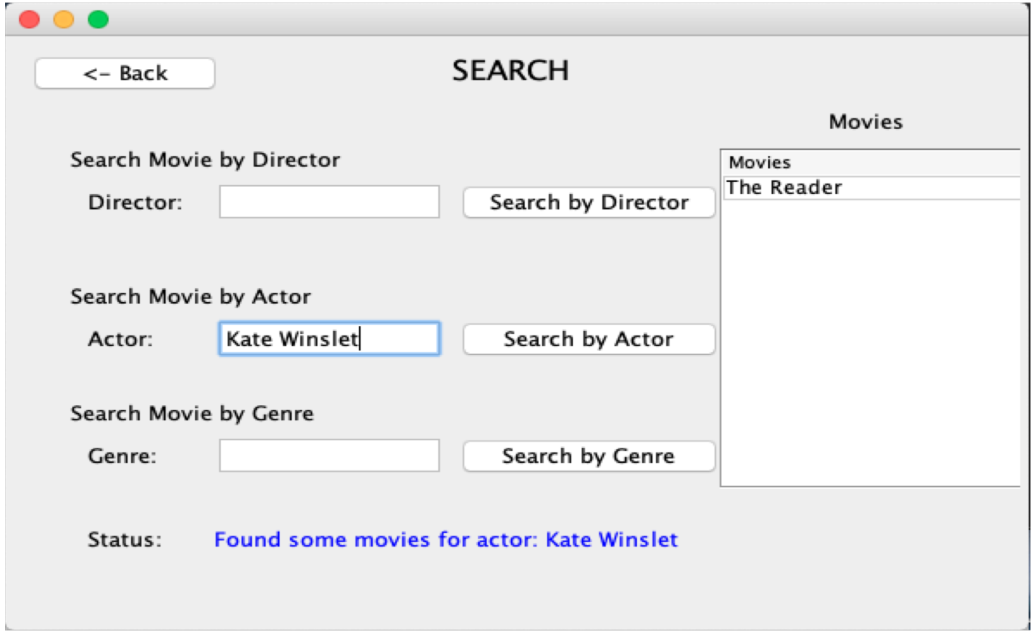
Movies

Movies
The Reader

Status: Found some movies for director: Stephen Daldry

Neo4j GUI Application: SEARCH BY ACTOR

Search Movie by Actor



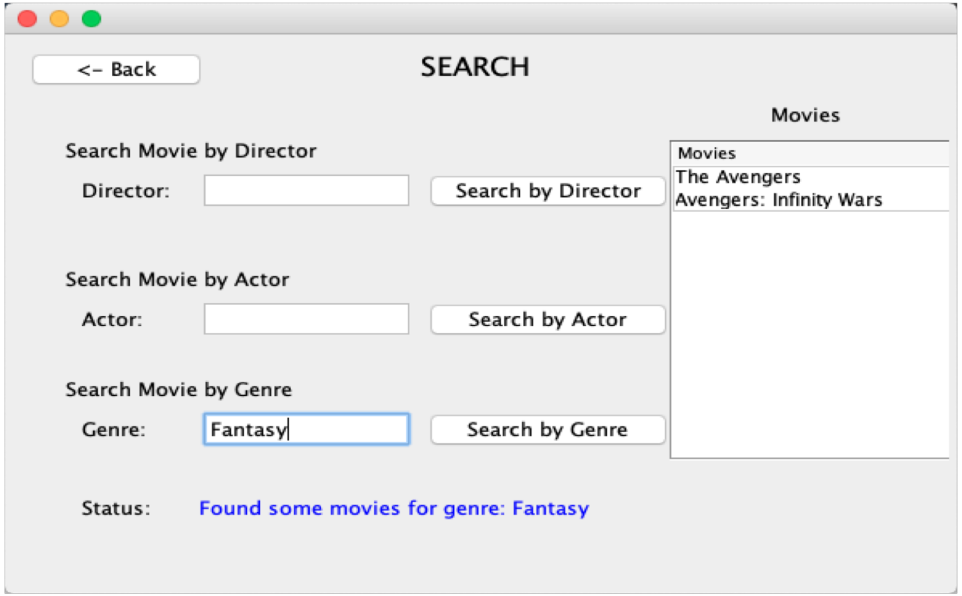
Actor: Kate Winslet

Movies
The Reader

Status: Found some movies for actor: Kate Winslet

Neo4j GUI Application: SEARCH BY GENRE

Search Movie by Genre



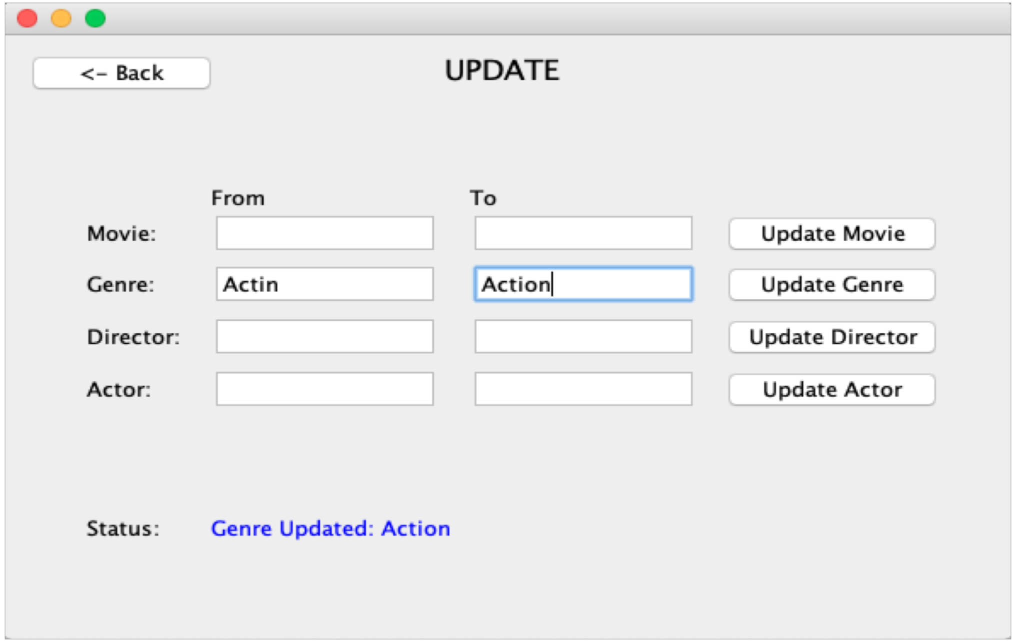
Genre: Fantasy

Movies
The Avengers
Avengers: Infinity Wars

Status: Found some movies for genre: Fantasy

Neo4j GUI Application: UPDATE GENRE

Update Misspelled Genre

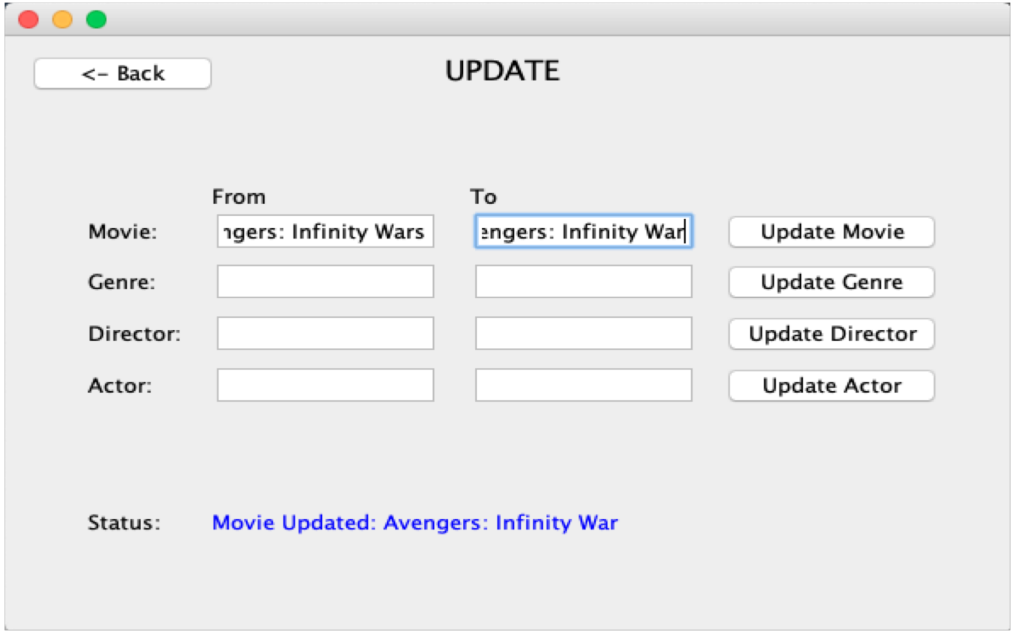


	From	To
Movie:		
Genre:	Actin	Action
Director:		
Actor:		

Status: Genre Updated: Action

Neo4j GUI Application: UPDATE MOVIE

Update Misspelled Movie

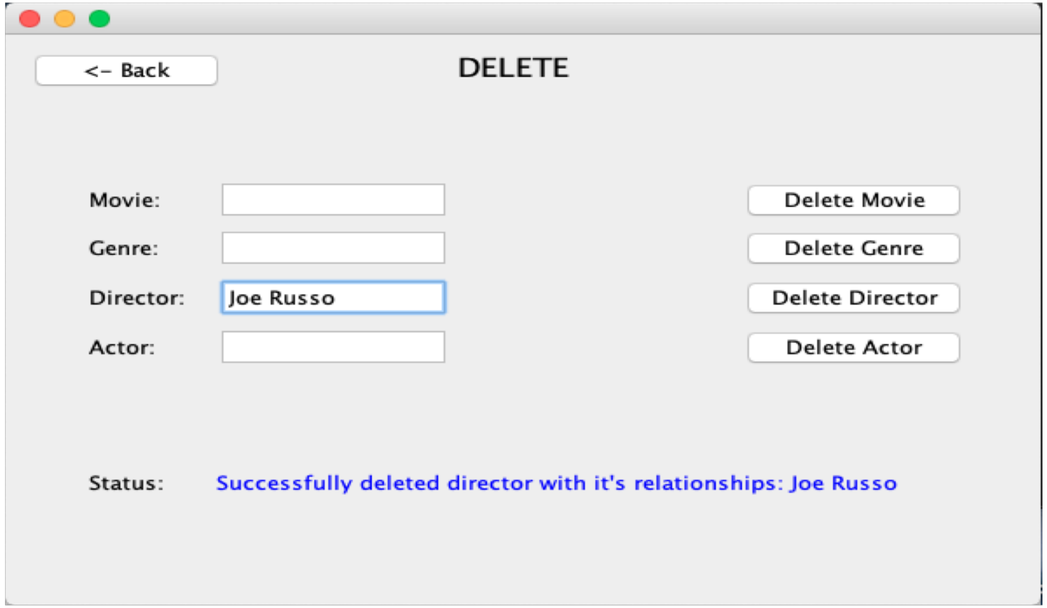


	From	To
Movie:	ɿgers: Infinity Wars	ɛngers: Infinity War
Genre:		
Director:		
Actor:		

Status: Movie Updated: Avengers: Infinity War

Neo4j GUI Application: DELETE DIRECTOR

Delete a Director



Movie:

Genre:

Director: Joe Russo

Actor:

Status: Successfully deleted director with it's relationships: Joe Russo

References

[1] E. Eifrem. Graph database service, 2002.
https://neo4j.com/docs/java-reference/current/javadocs/org/neo4j/graphdb/GraphDatabaseService.html

[2] E. Eifrem. Neo4j cypher refcard 3.4, 2002.
https://neo4j.com/docs/cypher-refcard/current/

[3] E. Eifrem. Neo4j cypher query language, 2002
https://neo4j.com/developer/cypher-query-language/

[4] E. Eifrem. Neo4j cypher query tuning, 2002
https://neo4j.com/docs/developer-manual/current/cypher/query-tuning/

[5] Neo4j. Chapter 7: Security. Accessed: 2018-10-26.

[6] I. Robinson, J. Webber, and E. Eifrem. Graph databases. O'Reilly Media, Inc., 2013.

Thank You