RIT | Golisano College of Computing and Information Sciences
School of Information

NEO4J PROJECT

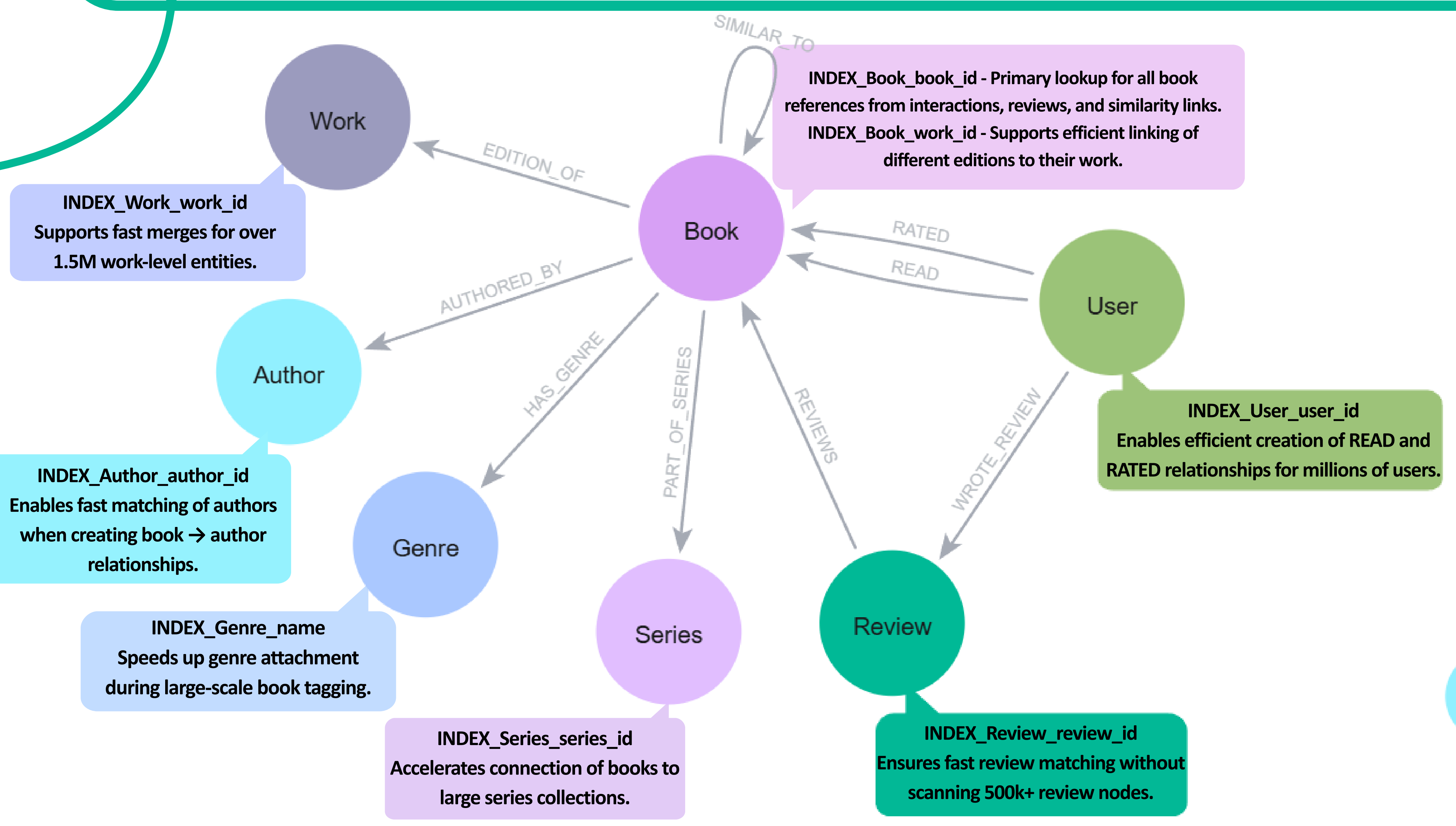
Goodreads Book Graph

Nodes 3,766,055

Relationship 11,425,891

Dataset

This project models Goodreads Comics & Graphic Novel data as a graph to explore complex relationships between books, users, authors, series, and reviews. A graph database was chosen to reveal hidden patterns in reading behavior, content similarity, and author influence.



INDEX_Book_book_id - Primary lookup for all book references from interactions, reviews, and similarity links. INDEX_Book_work_id - Supports efficient linking of different editions to their work.

INDEX_Work_work_id Supports fast merges for over 1.5M work-level entities.

INDEX_User_user_id Enables efficient creation of READ and RATED relationships for millions of users.

INDEX_Author_author_id Enables fast matching of authors when creating book → author relationships.

INDEX_Genre_name Speeds up genre attachment during large-scale book tagging.

INDEX_Series_series_id Accelerates connection of books to large series collections.

INDEX_Review_review_id Ensures fast review matching without scanning 500k+ review nodes.

Data loading

Data was imported from large JSON files using APOC procedures (apoc.load.json) with transactional batching. Nodes were created first with indexed properties, followed by relationship creation using foreign keys. Batch processing prevented memory overload and ensured efficient ingestion of millions of records.

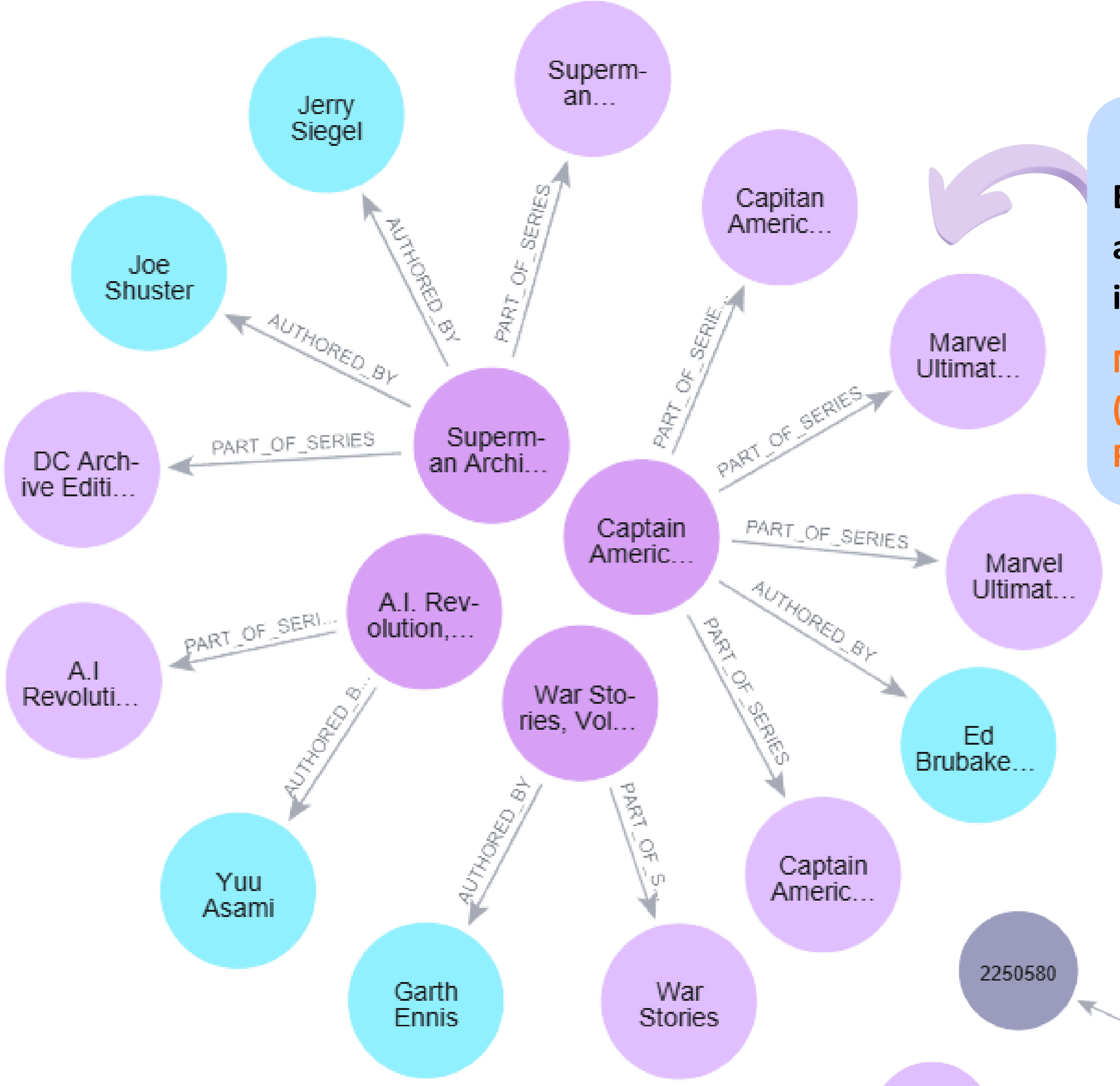
Why Neo4j?

Neo4j's native graph model is ideal for highly connected systems like Goodreads, where relationships are as important as data. It allows fast traversal across users, books, authors, genres, and reviews, which would be costly and complex in relational databases.

Performance

Indexing on key identifiers (book_id, user_id, work_id, etc.) and batch transactions enabled scalable performance. Compared to naïve loading, this approach significantly reduced import time and allowed complex multi-hop queries to run efficiently, even on a system with limited storage and memory.

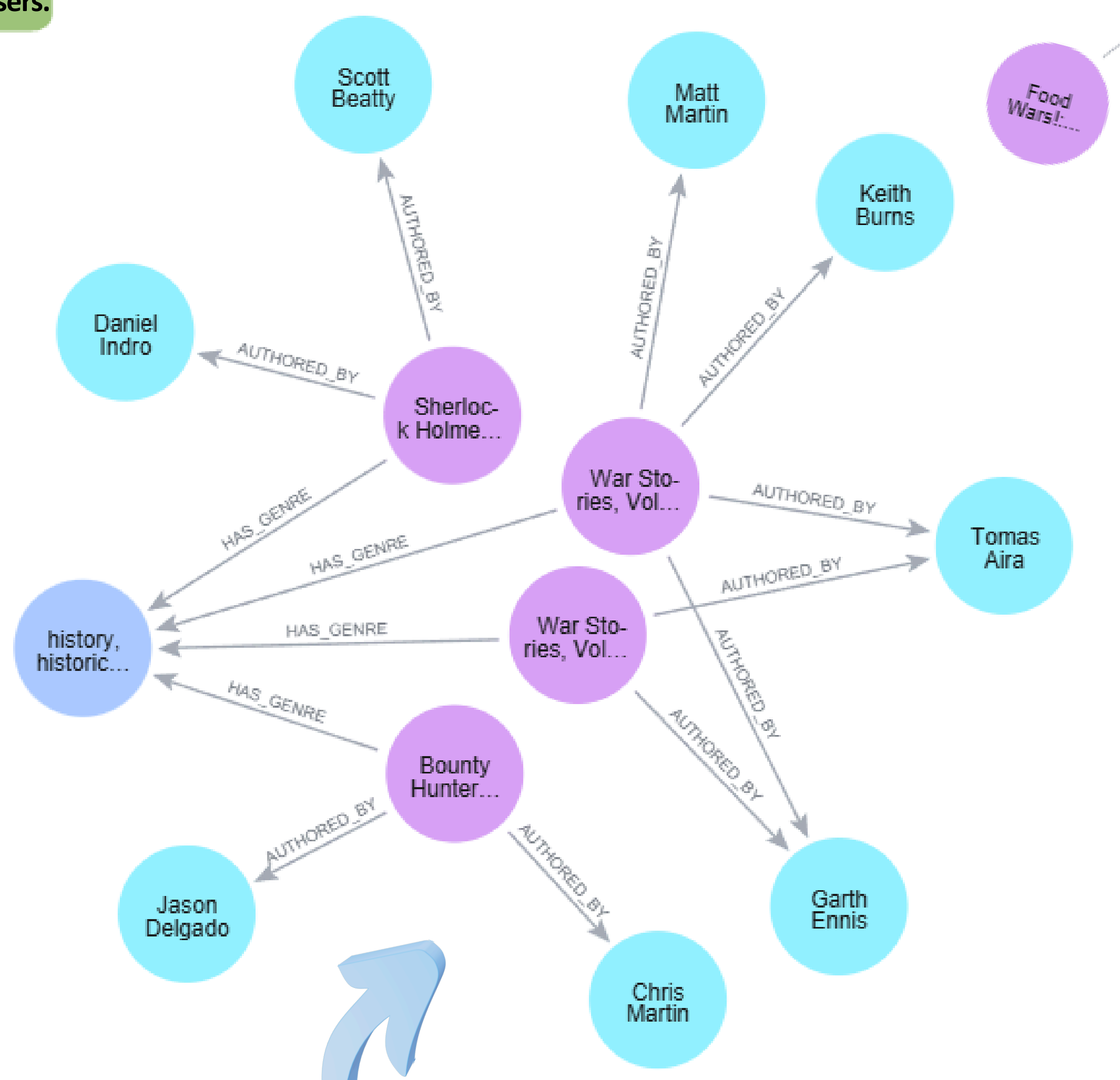
CYPHER QUERIES



Series → Book → Author

Explores how comic series are structured around specific authors, revealing author influence across interconnected book series.

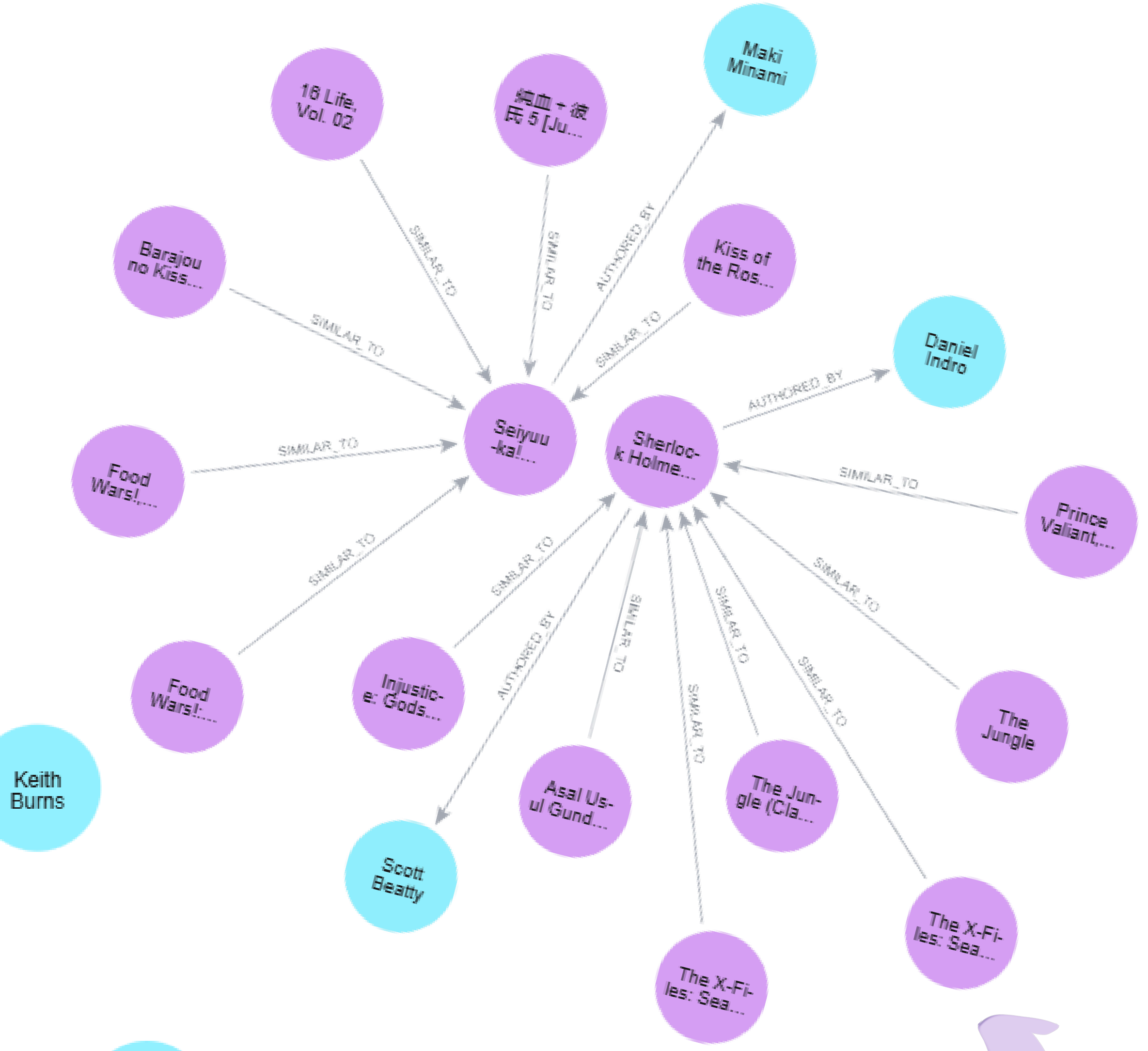
MATCH p = (s:Series)<-[:PART_OF_SERIES]-(b:Book)-[:AUTHORED_BY]->(a:Author)
RETURN p LIMIT 10;



Genre → Book → Author

Shows how different authors contribute to shaping genres through their published comic works.

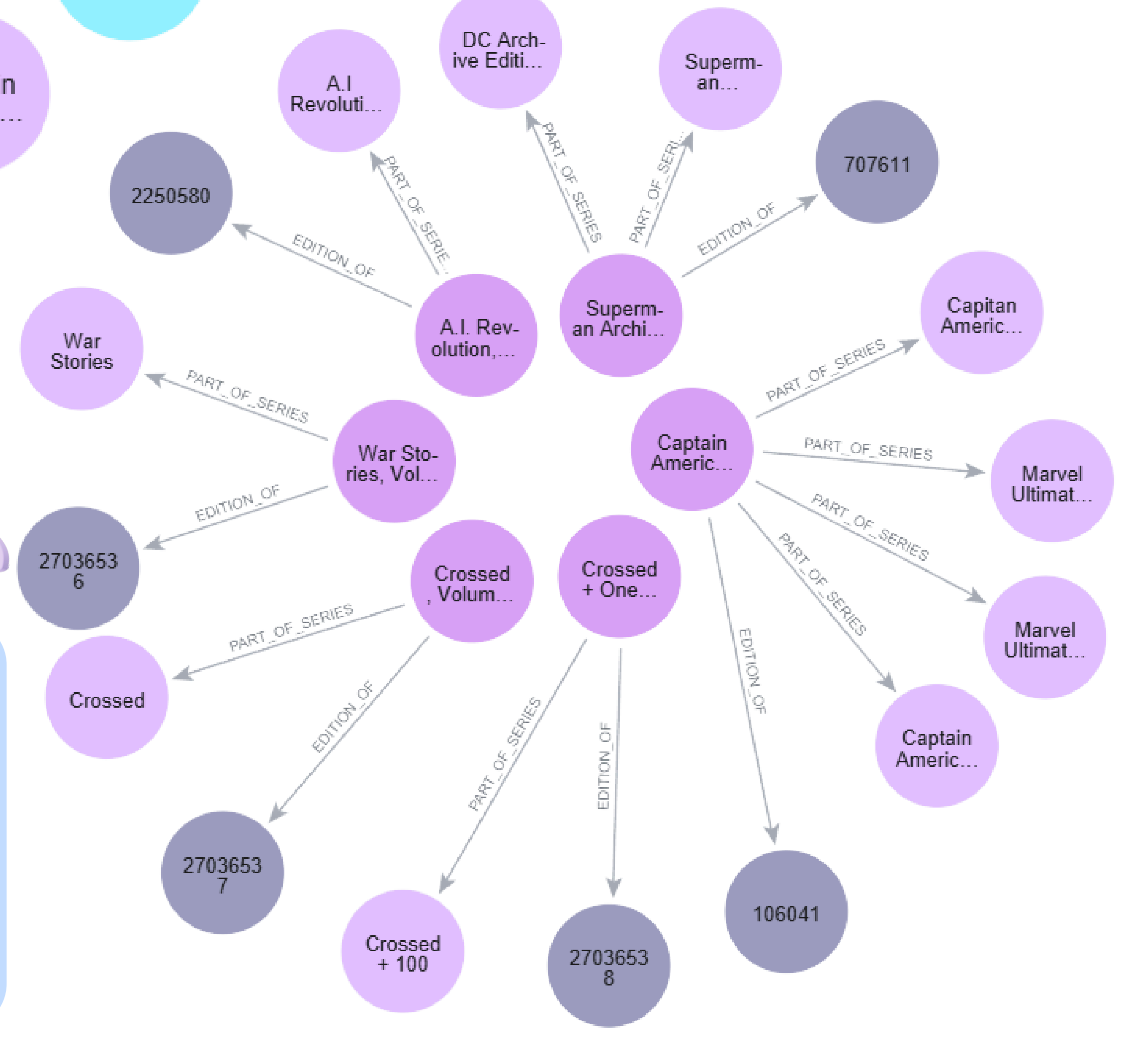
MATCH p = (g:Genre)<-[:HAS_GENRE]-(b:Book)-[:AUTHORED_BY]->(a:Author)
RETURN p LIMIT 10;



Book → Similar Book → Author

Reveals author ecosystems created through content similarity between comics, highlighting thematic or stylistic connections.

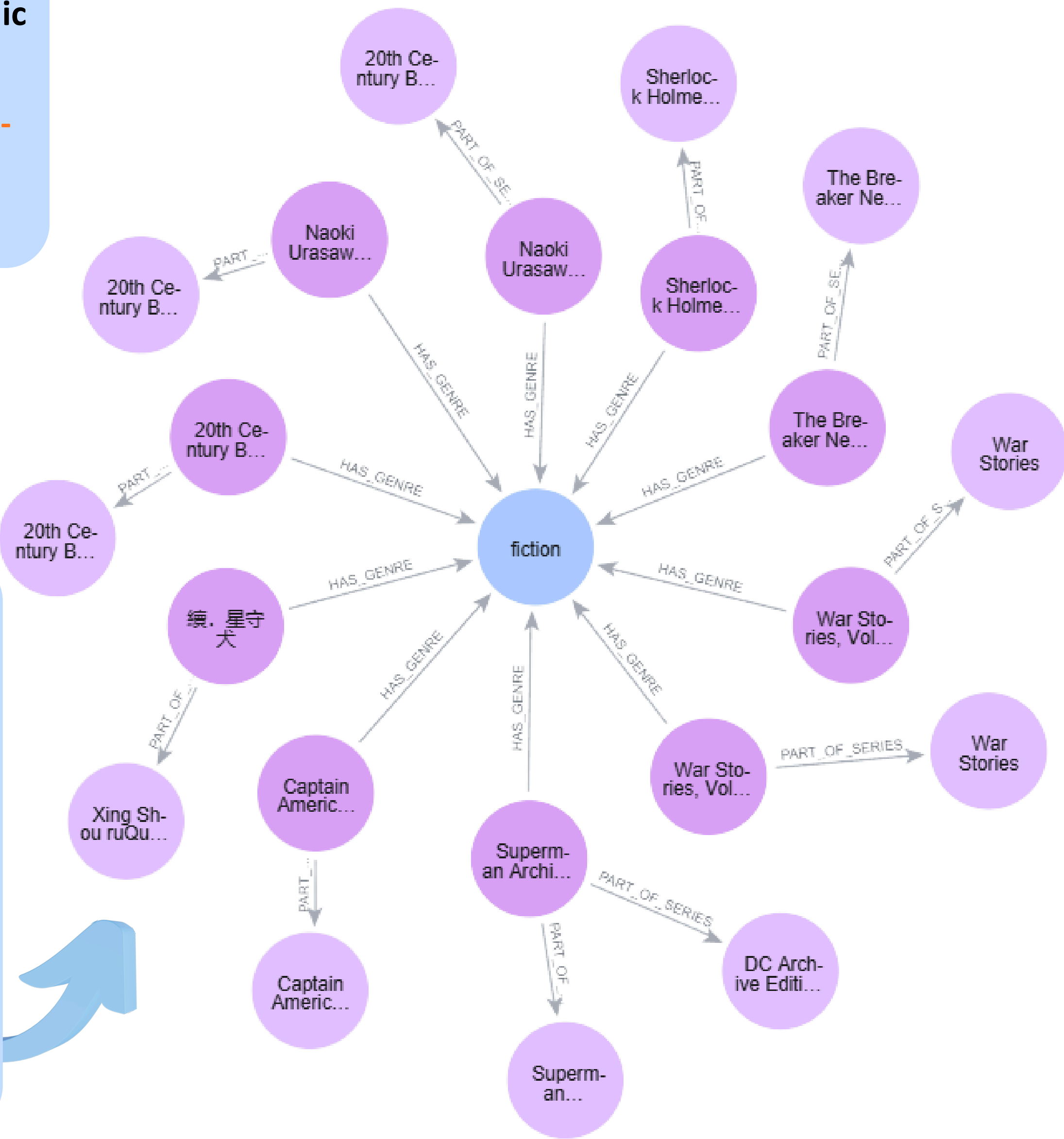
MATCH p = (b:Book)-[:SIMILAR_TO]->(b2:Book)-[:AUTHORED_BY]->(a:Author)
RETURN p LIMIT 20;



Work → Book → Series

Visualizes how multiple book editions of the same work are distributed across different series collections.

MATCH p = (w:Work)<-[:EDITION_OF]-(b:Book)-[:PART_OF_SERIES]->(s:Series)
RETURN p LIMIT 10;



Series → Book → Genre → Book → Series

Identifies how distinct comic series are indirectly connected through shared genres, exposing cross-series thematic networks.

MATCH p = (s1:Series)<-[:PART_OF_SERIES]-(b1:Book)-[:HAS_GENRE]->(g:Genre)<-[:HAS_GENRE]-(b2:Book)-[:PART_OF_SERIES]->(s2:Series)
WHERE s1 <> s2
RETURN p LIMIT 10;