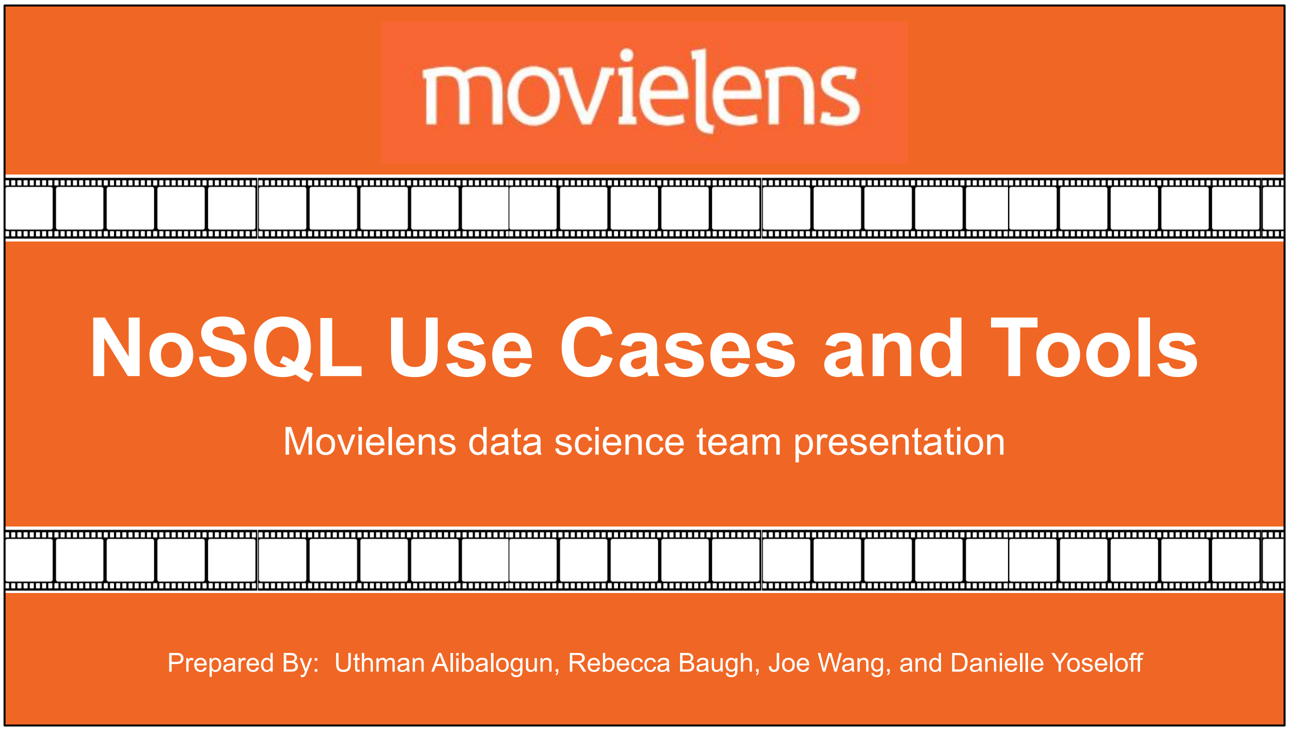


movielens

NoSQL Use Cases and Tools

Movielens data science team presentation

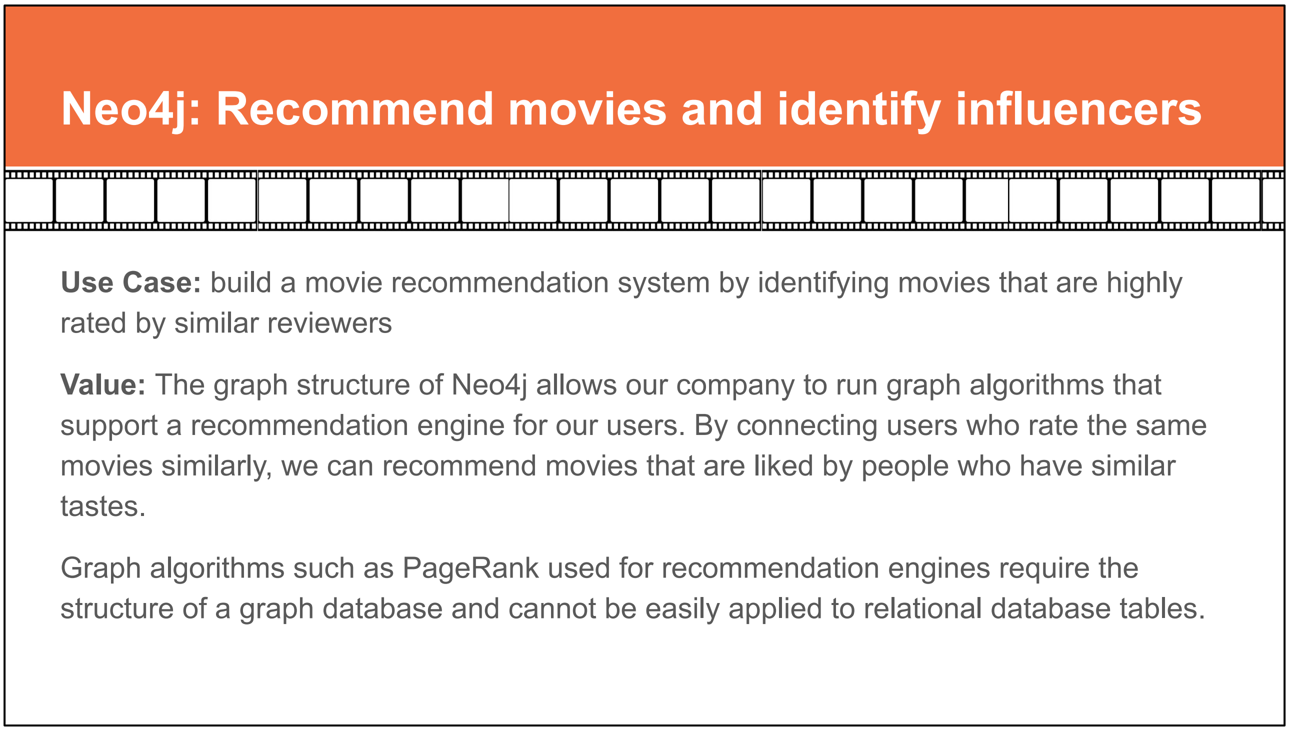
Prepared By: Uthman Alibalogun, Rebecca Baugh, Joe Wang, and Danielle Yoseloff

- Hello I'm Rebecca and together with Uthman, Joe, and Danielle we worked with movielens data. Movielens is a movie rating and reviewing community which publishes a lot of its data for academic purposes.
- Today, we are going to be presenting to the fictional movielens data science team, who has expressed interest in learning more about NoSQL databases. So from now on you are all movielens data-scientists.
- Welcome to today's lunch and learn

Topics

- **Neo4j**
  - Use Case: Recommend movies and identify influencers
  - Proof of concept using test data: PageRank
- **MongoDB**
  - Use Case: Allow users to tag favorite genres
- **Redis**
  - Use Case: Allow users to see real time updates to recommendations and top reviewers

- Today we are going to be covering three different potential tools.
- All of these tools support NoSQL databases and we will be talking about then in the context of how we can provide value to our users
- First we will cover a Neo4j use case and talk about some proof of concept tests we ran around recommendations
- Then we will move on to MongoDB and how we can help users tag their favorite genres
- Lastly we will talk about Redis and how to give users the most up to date information within each session



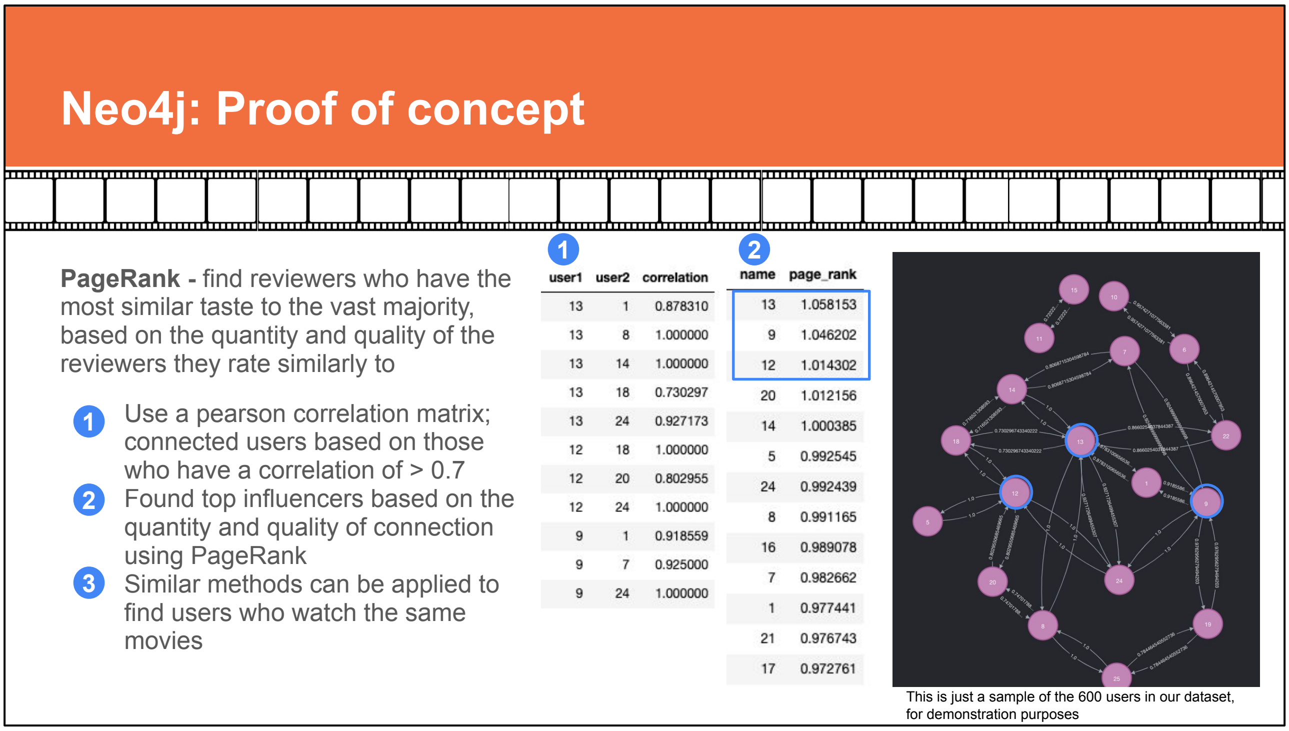
Neo4j: Recommend movies and identify influencers

Use Case: build a movie recommendation system by identifying movies that are highly rated by similar reviewers

Value: The graph structure of Neo4j allows our company to run graph algorithms that support a recommendation engine for our users. By connecting users who rate the same movies similarly, we can recommend movies that are liked by people who have similar tastes.

Graph algorithms such as PageRank used for recommendation engines require the structure of a graph database and cannot be easily applied to relational database tables.

- By connecting users who rate the same movies similarly, we can recommend movies that are liked by people who have similar tastes
- Currently we have an existing relational database of users with rating and movies. There are millions of users, thousands of movies, and intricate relationships between them.
- The relational database format is limited in its ability to analyze this kind of information, preventing effective recommendations
- Neo4j is a nosql graph database making it a great option to represent this information with all it's intricate relationships
- For Neo4j we have run a number of experiments on our companies data as a proof of concept.
- Joe will go into detail about the PageRank algorithm that provides insight into this data but I want to take a minute to explain the graph set up
- We can create each user as single node, and connect them with a correlation based on similar ratings
- Another way to connect user nodes is by the number of movies they have watched in common
- We explored both those options and were able to a make number of graph algorithms
- Now off to Joe go into the details



Neo4j: Proof of concept

PageRank - find reviewers who have the most similar taste to the vast majority, based on the quantity and quality of the reviewers they rate similarly to

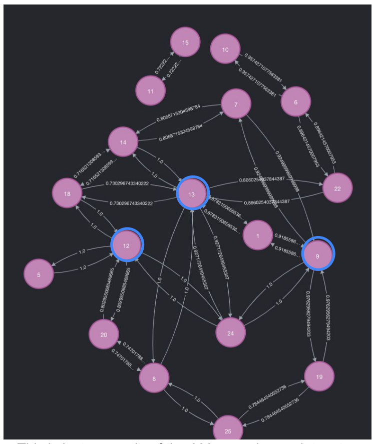
1. Use a pearson correlation matrix; connected users based on those who have a correlation of > 0.7
2. Found top influencers based on the quantity and quality of connection using PageRank
3. Similar methods can be applied to find users who watch the same movies

1

user1	user2	correlation
13	1	0.878310
13	8	1.000000
13	14	1.000000
13	18	0.730297
13	24	0.927173
12	18	1.000000
12	20	0.802955
12	24	1.000000
9	1	0.918559
9	7	0.925000
9	24	1.000000

2

name	page_rank
13	1.058153
9	1.046202
12	1.014302
20	1.012156
14	1.000385
5	0.992545
24	0.992439
8	0.991165
16	0.989078
7	0.982662
1	0.977441
21	0.976743
17	0.972761



This is just a sample of the 600 users in our dataset, for demonstration purposes

Another way for Movielens to increase their number of users is through influencers in the form of experienced reviewers

The problem is these reviewers have diverse taste, it is hard to know if the reviewer has the same preference as you

PageRank can find reviewers who have the most similar taste to the vast majority, based on the quantity and quality of the reviewers they rate similarly to

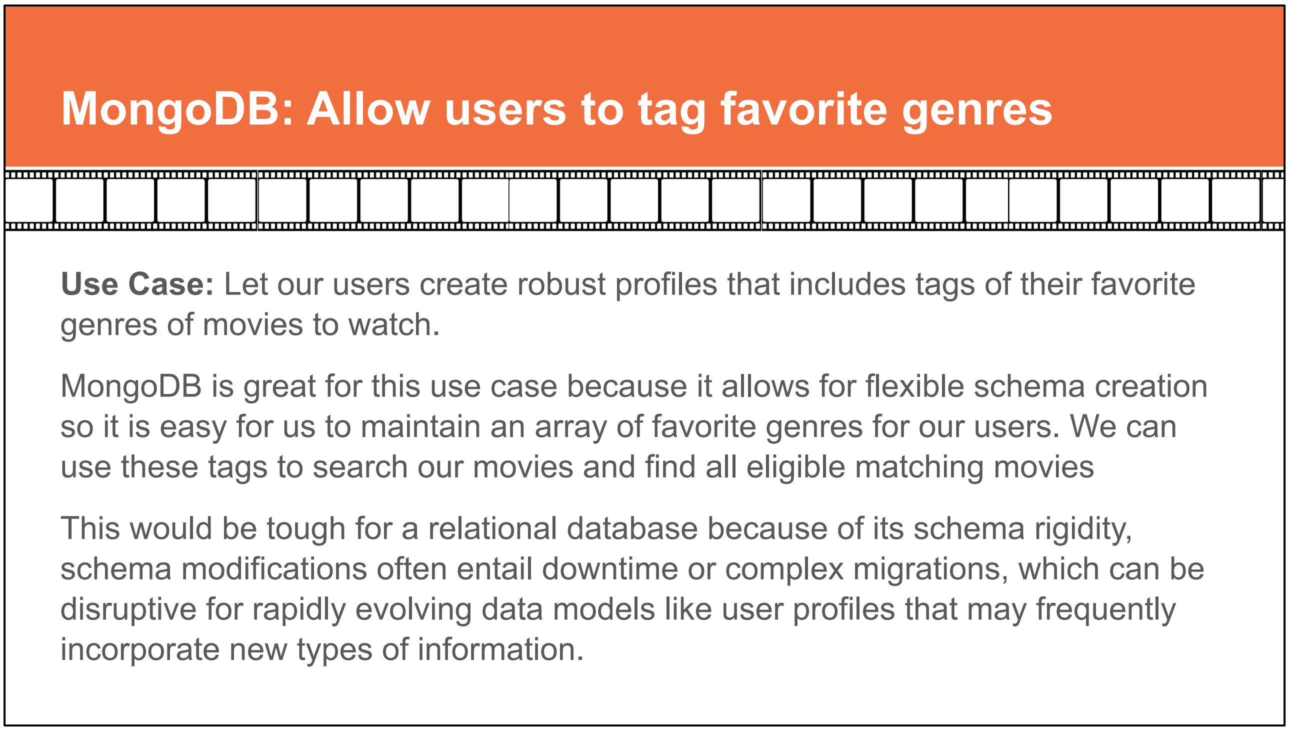
To do this, we first created a correlation matrix of 600+ Movielens reviewers, based on similar review patterns of the same movies.

We then built connections between users who are highly correlated in Neo4J, with their relationship weighted by their correlation coefficient.

You can see a simplified version of this on the first chart to the right, where we have two nodes; user 1 and user 2. This is a one to many relationship. Where one user could be connected to multiple other users; based on a correlation more than 0.7.

page rank results: where users 13, 9, 12 have the highest score because of the quantity and quality of their connections with other users you see in the first table. You can then see this visualized in the graph to the far right.

Pagerank is just the beginning. For example, we can run louvain modularity to create community groups within our users; so we can do a “users who rated highly on movie x also rated highly on movie y that you might like” for MovieLens users



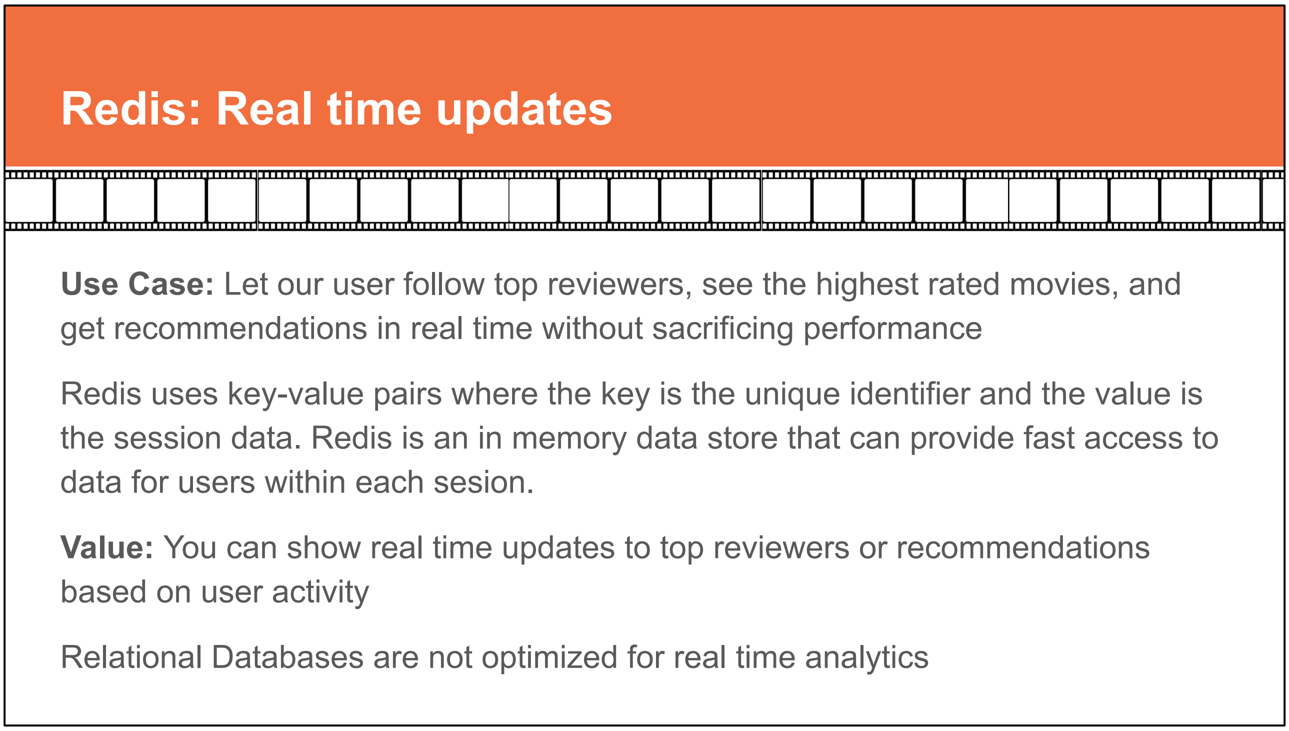
MongoDB: Allow users to tag favorite genres

Use Case: Let our users create robust profiles that includes tags of their favorite genres of movies to watch.

MongoDB is great for this use case because it allows for flexible schema creation so it is easy for us to maintain an array of favorite genres for our users. We can use these tags to search our movies and find all eligible matching movies

This would be tough for a relational database because of its schema rigidity, schema modifications often entail downtime or complex migrations, which can be disruptive for rapidly evolving data models like user profiles that may frequently incorporate new types of information.

- Framing this in a business context, we want to allow our users to create robust dynamic profiles where they can personalize their experience
- If we want to allow users to tag their favorite genres, we need to allow for changes in genres.
- MongoDB is a document database where data is stored in field value pairs, we can use this structure to let uses set tags or keys where we can match movies to their profile based on fit.
- The document data structure allows for flexibility within these field value pairs, meaning we can expand to different genres or different tagging features as needed.
- We don’t want to have to update rigid relational database schemas every time we want to add new personalization features to our users profile
- MongoDB would allow us to rapidly adapt to different tags our users were interested in, surfacing the preferred genres in on their profiles
- The main value of MongoDB is the ability for us to be able to change with our users without large scale migrations or lagging systems
- This type of data structure sets us up for success as we try to expand our user base and our functionality



Redis: Real time updates

Use Case: Let our user follow top reviewers, see the highest rated movies, and get recommendations in real time without sacrificing performance

Redis uses key-value pairs where the key is the unique identifier and the value is the session data. Redis is an in memory data store that can provide fast access to data for users within each sesion.

Value: You can show real time updates to top reviewers or recommendations based on user activity

Relational Databases are not optimized for real time analytics

- Moving on to our next to Redis, before we think about the tool let's think about our user experience
- We want our users to be able to follow top reviewers, see the highest rated movies, and get recommendations based on real time data.
- Redis can help us provide that experience without sacrificing performance and bringing down our production environment
- Redis uses key-value pairs where the key is the unique identifier and the value is the session data.
- Redis is an in memory data store so it can process the volume and velocity of data we need in order to let users know who top reviewers are and what the best movies with a live leaderboard.
- Redis also can use their most recent session data to provide the best recommendations based on their reviews as Joe discussed.
- This session data is not limited to giving real time dashboards, after the user completes their session we can pass the session data to a different database where we can run additional analytics as needed.
- If we tried to give this sort of personalized experience with a relational database we would bring down our production environment.
- We are not saying that we need to get rid of all relational databases
- Today we simply wanted to give an overview of some experiences we could provide to our users by leveraging NoSQL databases in addition to our existing relational databases.
- Thank you for your time and please let us know if you have any questions.

Thank You

•