SMRSEA

Search

Home

Library

Our Music Playlists

Smart Music Recommendation System for Emerging Artists

A fairer recommendation system based on user behavior and music similarity

Business Case Scenario

Popular streaming platforms **prioritize mainstream artists** , making it hard for emerging musicians to get discovered.

This project creates **a fair recommendation system** based on user behavior and music similarity by using graph databases to identify relationships in music preferences to **prioritize emerging artist recommendations.**

Dataset Features

SMRSEA

User ID

1h4min ▶

a hash of the user's Spotify username

Artist

45min ▶

the name of the artist

Playlist

30min ▶

name of the playlist containing the song

Song

1h15min ▶

the title of the track

Search

Home

Library

Our Music Playlists

SMRSEA

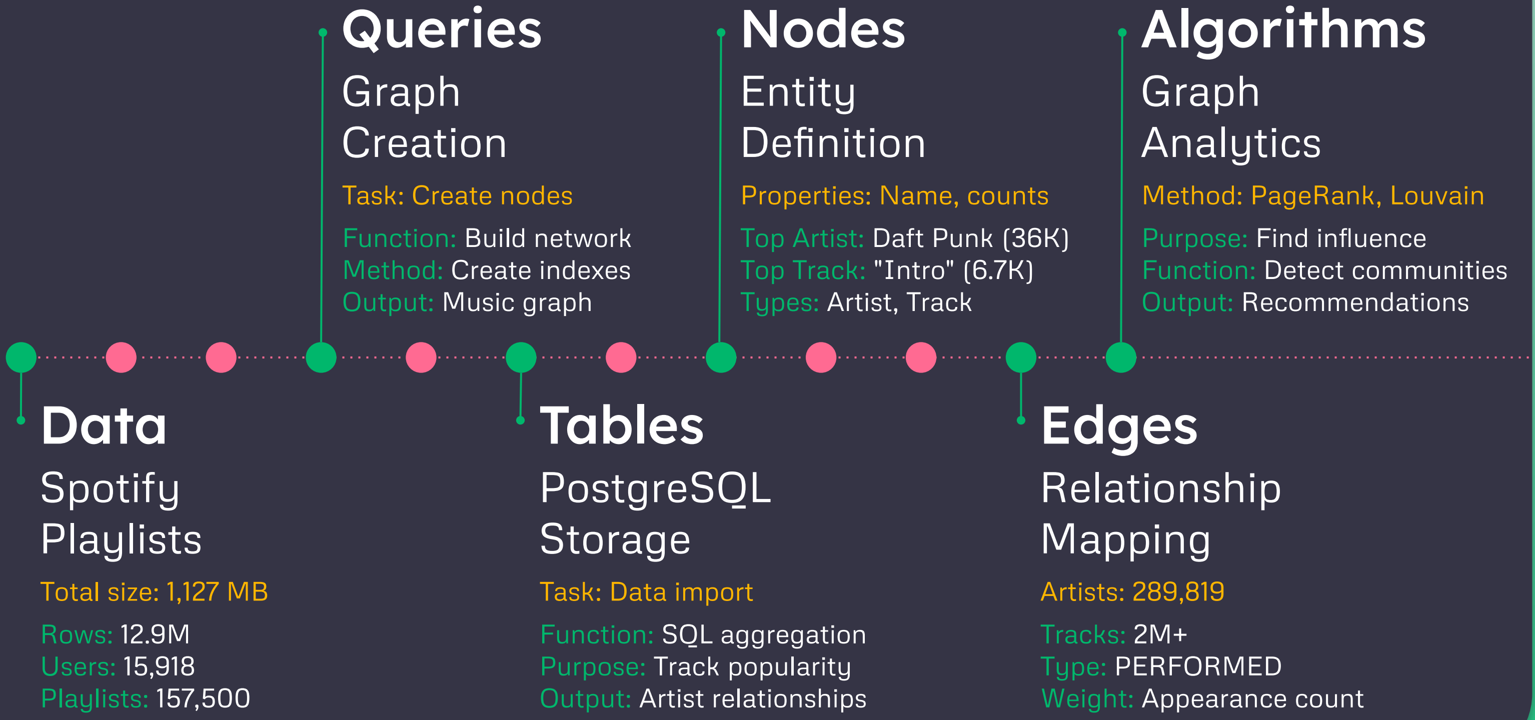
Neo4j
use case

Follow

Our recommendation system uses Neo4j to model music as a network of connected users, artists, and tracks. It uncovers hidden patterns to identify emerging artists and create recommendation paths.

- A relational database would be slower and less efficient.

Our Approach



Data

Spotify Playlists

Total size: 1,127 MB

Rows: 12.9M
Users: 15,918
Playlists: 157,500

Queries

Graph Creation

Task: Create nodes

Function: Build network
Method: Create indexes
Output: Music graph

Tables

PostgreSQL Storage

Task: Data import

Function: SQL aggregation
Purpose: Track popularity
Output: Artist relationships

Nodes

Entity Definition

Properties: Name, counts

Top Artist: Daft Punk (36K)
Top Track: "Intro" (6.7K)
Types: Artist, Track

Edges

Relationship Mapping

Artists: 289,819

Tracks: 2M+
Type: PERFORMED
Weight: Appearance count

Algorithms

Graph Analytics

Method: PageRank, Louvain

Purpose: Find influence
Function: Detect communities
Output: Recommendations

Nodes

SMRSEA

● Artist

Follow

Total of 289,819

Connected through co-listening patterns

● Track

Follow

2M+

Average track appears in 6-7 playlists

Each track connects to exactly one artist

● User

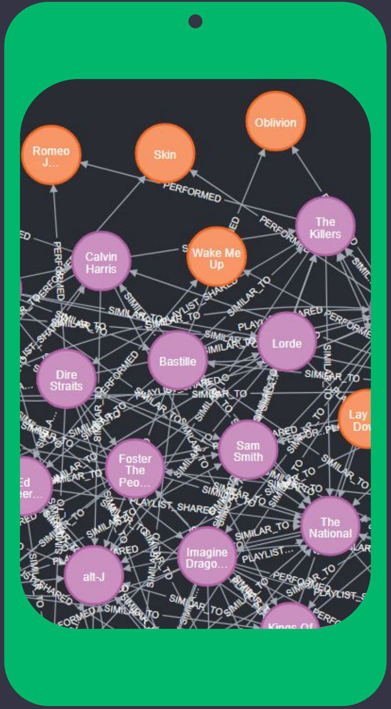
Follow

Average of 206 unique artists per user

Average of 705 unique tracks per user

14-15 playlists per user on average

SMRSEA



Edges:

number of playlists

PERFORMED (Artist →Track)

- Links 289K artists to 2M+ tracks

SIMILAR_TO (Artist ↔Artist)

- Created from co-listening patterns

APPEARS_IN (Track →Playlist)

- Average playlist: 56 tracks
- Maximum playlist size: 47,362 tracks

Neo4j Graph Algorithms

SMRSEA

PageRank

3:15

Identifies influential artists in the music network

Louvain Community Detection

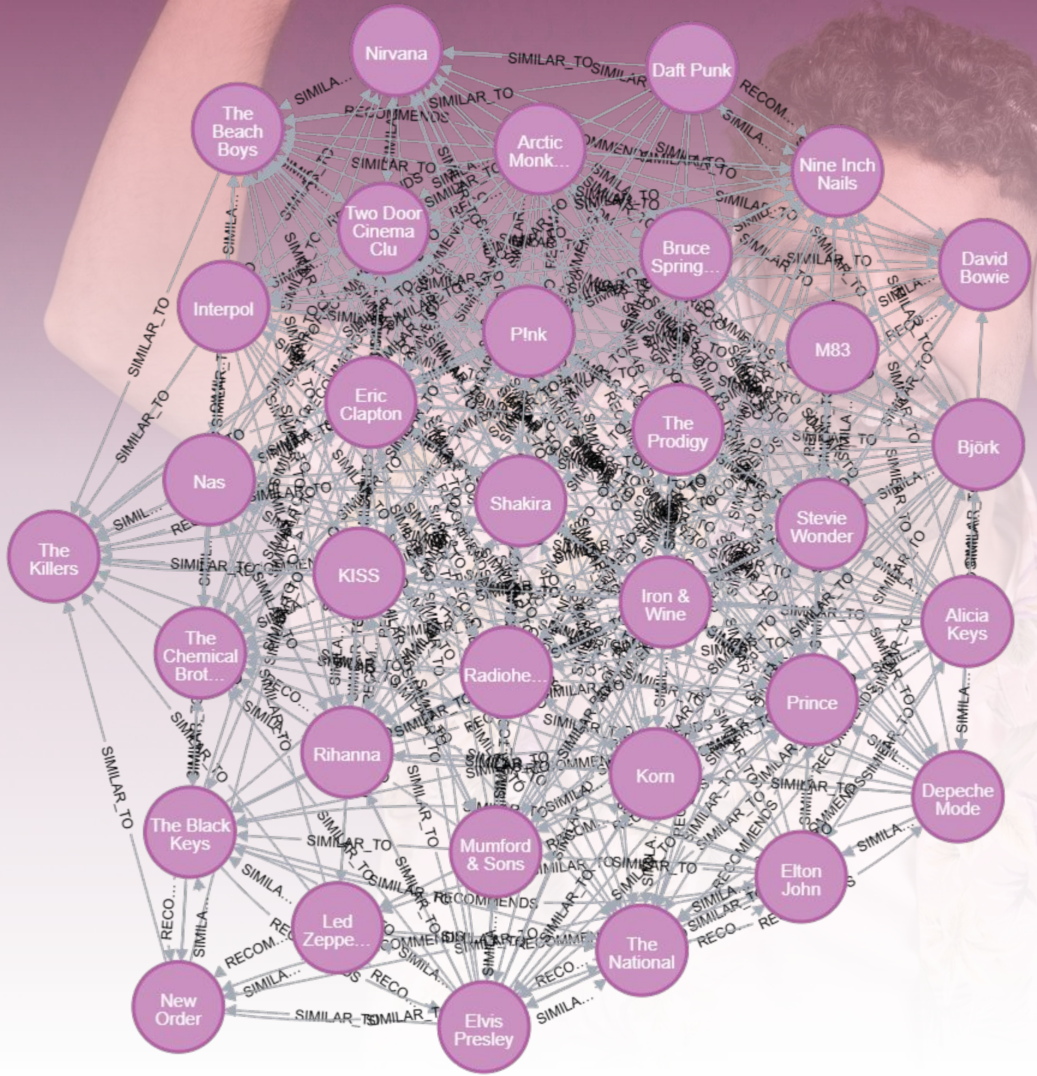
3:20

Discovers natural music communities/genres

Multiple similarity calculations

3:10

Combines shared tracks (cosine similarity) and genre overlap (Jaccard)



Neo4j graph Similarity Relationships

SMRSEA

Search

Home

Library

Our Music Playlists

SMRSEA

MongoDB

Follow

use case

Storing artist metadata with varying attributes to support flexible searches and identifying emerging artists

MongoDB vs. Relational DB

SMRSEA

MongoDB

Strengths

- Nested data (track features and collaboration lists)
- Schema flexibility supports evolving artist data without migration
- Real-time analytics and filtering

Relational DB

Limitations

- Joins between artists, tracks, and genres ➪ complex
- Schema rigidity

Search

Home

Library

Our Music Playlists

SMRSEA

Redis use case

Follow

Storing and retrieving recently played artists, session data, and trending emerging artists in real time

Redis vs. Relational DB

Redis

Strengths

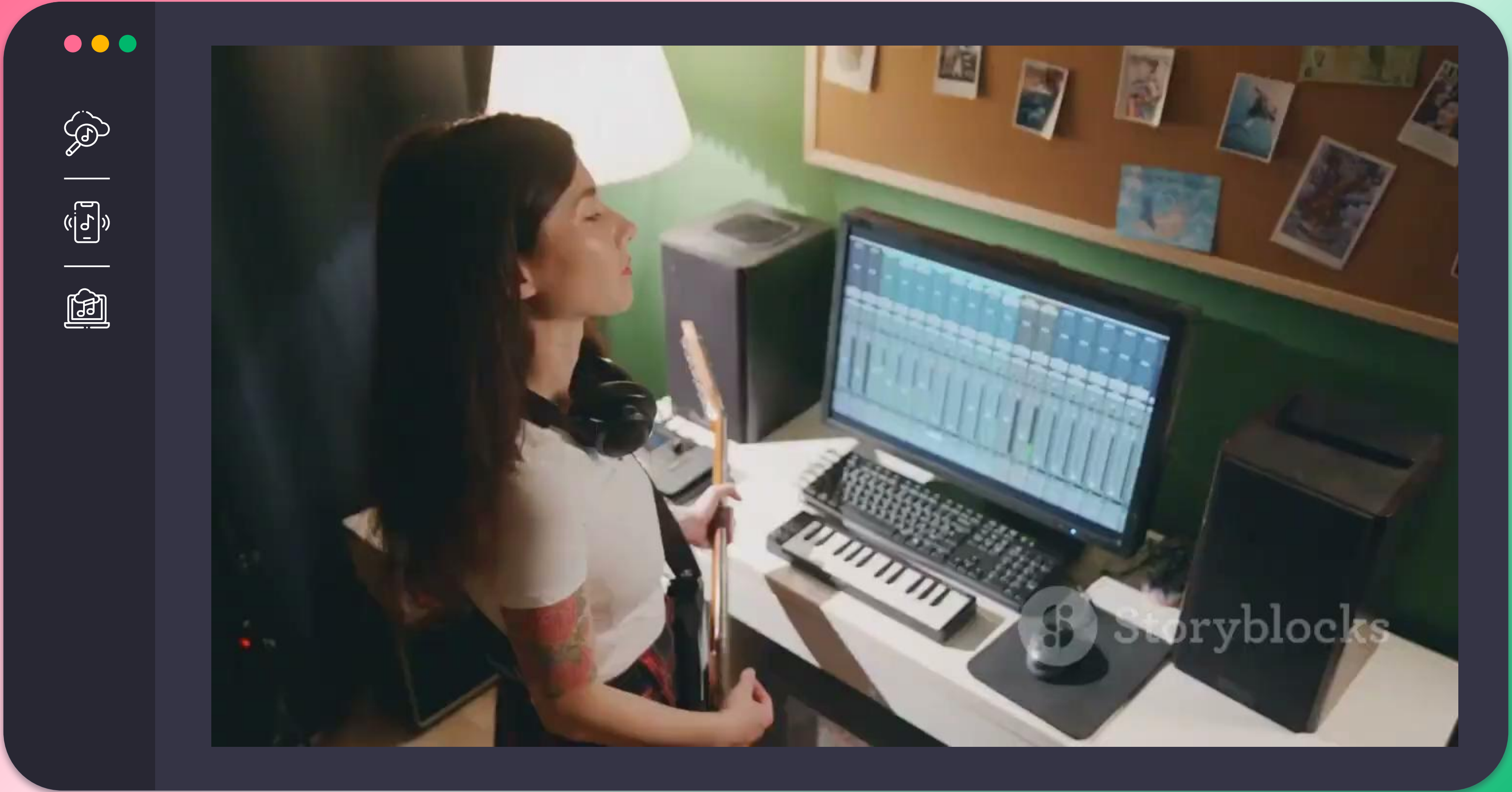
- Real-time access + in-memory storage
- Frequent updates
- Ideal for fast-changing data
- Optimized key-value format for speed and scalability

Relational DB

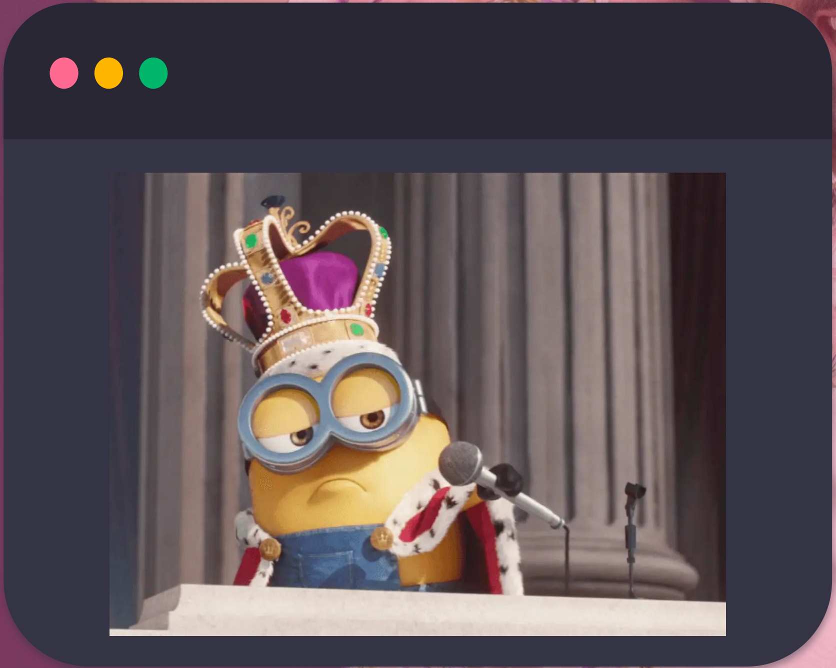
Limitations

- Slow write/read speeds
- Live activity tracking or session caching is difficult
- Complex schemas

SMRSEA



SMRSEA



Thank you!

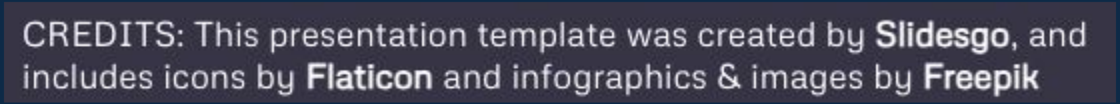
Citations

Redis Image:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.tothenew.com%2Fblog%2Fredis-a-comprehensive-guide%2F&psig=AOvVaw2q6bW_ZlT2OVLeEn6lTQjD&ust=1743983690681000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCLiOndmLwowDFQAAAAAdAAAAABAE

MongoDB Image:
https://www.google.com/url?sa=i&url=https%3A%2F%2Ftwitter.com%2FMongoDB&psig=AOvVaw2wu63VEDTxEPVA6ifizbVh&ust=1743983564878000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCMDXn52LwowDFQAAAAAdAAAAABAd

neo4j image:
https://www.google.com/search?sca_esv=9d0194921f740f64&rlz=1C5CHFA_enUS939US939&q=neo4j+logo&uds=ABqPDvztZD_Nu18FR6tNPw2cK_RRLt3M0EtvWqtCZ6tbVcLtuPGD0a4v_ftS7KAOqrr0clJe3SgYh8keeiVCFBScZyYG6NRSx014NChT6IAoekaIoRxx92rpBSYr1R_itPvN1wfBGzRv&udm=2&sa=X&ved=2ahUKEwijvcW4jsKMAxVsJNAFHf6MCwcQxKsJegQIDBAB&ictx=0&biw=1920&bih=958&dpr=1#vhid=vK-xSSkvB3UivM&vssid=mosaic

Kevin video:
https://bcourses.berkeley.edu/courses/1540223/pages/2-dot-3-1-concepts-relational-databases?module_item_id=17262685



CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon** and infographics & images by **Freepik**