#ASLI ENGINEERING

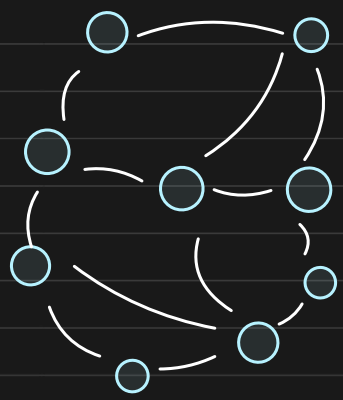
Architecture of Airbnb's Knowledge Graph

SWIPE

BY

ARPIT BHAYANI

Airbnb's knowledge graph

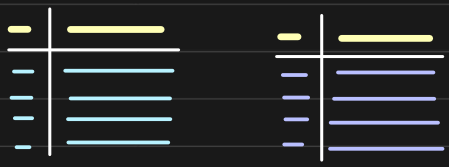


Airbnb holds experiences, places, hotels, events, restaurants, markets, homes, and much more.

While planning a trip, it is important to show all critical information to the user for making an optimal decision.

This information is stored in a Knowledge Graph

Can we not store it in Relational DB?



- most common storage
- one row for each entity
- linkage through Foreign Key references

Relational Databases work well with transactional data where we want to access certain rows, but

- find city that hosts a type of experience in July and August
- find neighbourhood in LA where there are Huts or Islands available

Answering such queries is a big pain with relational databases

We want to 'surf' within the data....

Infrastructure

Knowledge Graph infrastructure has 3 key components

- Graph Storage
- Graph Query API
- Storage Mutator

Graph Storage

The entire knowledge Graph is stored in a relational database, but it is structured as Nodes and Edges as tuples

< subject , verb , object >

eg: NewYork , IS_IN , USA

Marriot-17 , IS_IN , USA

reference id

Each node type has a diff schema

eg: location → Name, GPS coord

Event → Name, Date, Venue

Each edge type stores nodes that connect

eg: landmark-in-city → landmark & place

Why didn't they use Graph DB?

Operation Overhead. Expertise

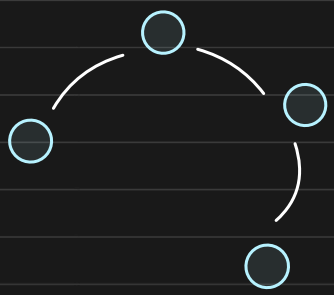
Knowledge graph is periodically dumped for offline consumption eg: recommendation / ranking

Graph Query API

Traverse the graph by specifying the path.

Path is just a sequence of edges

+ data filters



eg: [landmark-in-city , city-with-mountain]

filter : location : USA

eg: Find all 'place' nodes connected with 'city' node LA

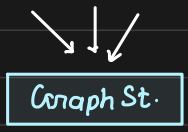
with edge type 'contains_location'

such that - # listing > 5000 - category = 'scenic'

Storage Mutator

Say, some information in the graph needs to be changed

So, we can expose an API endpoint to do so! Simple..



Not really!

Doing large number of synchronous mutations is

1. slow 2. expensive 3. time consuming

Hence, it is better to hunt for an 'async' alternative

Along with sync API updates,
the mutation can be sent over Kafka making the flow efficient