**Báo cáo,đánh giá tổng quan Dgraph và JanusGraph**

Đồ thị là một tập các đối tượng gọi là đỉnh nối với nhau bởi các cạnh. Thông thường, đồ thị được vẽ dưới dạng một tập các điểm (đỉnh, nút) nối với nhau bởi các đoạn thẳng (cạnh). Tùy theo ứng dụng mà một số cạnh có thể có hướng.

Graph Database mô tả và lưu trữ dữ liệu dưới dạng đồ thị, một cách trực quan và dễ dàng truy vấn. Dĩ nhiên Graph Database cũng sẽ có những ưu, khuyết điểm, những trường hợp nên và không nên sử dụng.

1. Dgraph
   1. Giới thiệu

Dgraph là một cơ sở dữ liệu đồ thị nguyên gốc, giao dịch, phân tán, nguồn mở.

* 1. Đặc điểm
     1. Schema
* Sử dụng cấu trúc free-schema
* Schema types đủ dùng
* Querying schema

schema { type index reverse tokenizer list count upsert lang }

* Các node được liên kết với nhau thông qua 1 trường trong node. Trường đó sẽ định nghĩa kiểu liên kết cho các node. Có thể có kết nối tới 1 node hoặc nhiều node
* Các cạnh biểu thị đồ thị có hướng.
  + 1. Query
* Sử dụng ngôn ngữ GraphQL+- là một ngôn ngữ query được phát triển trên ngôn ngữ truy vấn GraphQL.
* Câu lệnh truy vấn ngắn ,dễ hiểu,dễ thao tác
* Các chức năng :

+ Filter

+ Aggregation

+ Sort

+ Count

+ Limit

+ First,after,offset

+ Groupby

* + 1. Recommend

- Nên sử dụng trong các bài toán ít loại node và không cần chi tiết hóa các kết nối

- Thông tin chủ yếu trung tâm ở các đỉnh

\*Ưu điểm

- Có thể thông qua Graph UI để chỉnh sửa schema linh hoạt

- Dgraph supports GraphQL-like query syntax, and responds in JSON and Protocol Buffers over GRPC and HTTP.

- Cộng đồng sử dụng lớn (Slack)

- Tạo trực quan hóa :

Việc trực quan hóa dễ dàng hơn thông qua việc sử dụng DGraph UI.

\*Nhược điểm

- Hiện tại chưa tìm ra

1. JanusGraph
   1. Giới thiệu

JanusGraph là một cơ sở dữ liệu đồ thị phân tán, nguồn mở trong Quỹ Linux. JanusGraph có sẵn theo Giấy phép Apache 2.0. Dự án được hỗ trợ bởi IBM, Google và Hortonworks.

* 1. Đặc điểm
     1. Schema

- Sử dụng theo cấu trúc được định nghĩ từ trước.

- Xác định các cột để đánh index tăng tốc độ truy vấn và tìm kiếm

- Để xác định mối quan hệ giữa các node thông qua nhãn cạnh. Cạnh có thể xác định trước hoặc không . Lý do cần phải xác định cạnh trước để xác định và đánh index tăng tốc độ tìm kiếm và truy vấn.

- Cạnh xác định đồ thị có hướng . Vì thế 2 node để liên kết qua lại với nhau sẽ có 2 nhãn xác định riêng biệt

* + 1. Query
* Sử dụng trình truy vấn của Gremlin
* Truy vấn Gremlin là một chuỗi các hoạt động / chức năng được đánh giá từ trái sang phải.
* Truy vấn dài và dễ nhầm lẫn
* Các chức năng :

+ Filter

+ Groupby

+ Count

+ Sort

+ Limit

+ Tail

* + 1. Recommend
* Xác định chi tiết các mối quan hệ
* Các bài toán liên quan ,tập trung xác định thông tin các cạnh
* Số bạn xung quanh
* Tìm kết nối ngắn nhất giữa 2 đỉnh

\*Ưu điểm

- Được sự hỗ trợ bởi  IBM, Google và Hortonworks.Do vậy IBM đã tạo SaaS dựa trên JanusGraph

- Xác định chi tiết mối quan hệ (social network)

\*Nhược điểm

- Không có UI để quản lý. Quản lý hoàn toàn bằng các câu lệnh

- Trực quan hóa cần phải sử dụng phần mềm của bên thứ 3

- Cộng đồng hoạt động không thường xuyên (gitter,groups.google)

1. Tổng quan

|  |  |  |
| --- | --- | --- |
| Graph | Dgraph | JanusGraph |
|  |  |  |
| Ngôn ngữ phát triển | Go | Java |
| Data Schema | Schema-free | có |
| Ngôn ngữ truy vấn | [GraphQL+-](https://docs.dgraph.io/tips/) | Query Gremlin |
| Ngôn ngữ hỗ trợ | C#  C++  Go  Java  Python(pydgraph)  Javascript  PHP  Ruby | Clojure  Java  Python(gremlinpython) |
| Trực quan hóa | DGraph UI | Gephi(GraphStreaming) Cytoscape(lib)  KeyLine |