Stats and reports data:

Can be any form of database either sql or nosql

Here we can have all the stats data and reports data.

RDF –triplet’s database:

We have our ontology and taxonomy in this db

Why do a separate db:

* We can opens source our taxonomy and ontology.
* We can use the any tool to build this taxonomy and ontology structure
* Or build a Ui layer on top to create a taxonomy structure.

We have three DB

* + - Apache jena , Allegro Graph , GraphDB
    - All three have proper integration with RDF triple storage and follow W3C stands for ontology and taxonomy

Crawled Data:

All the data crawled ca be dumped in to a data lake or big data

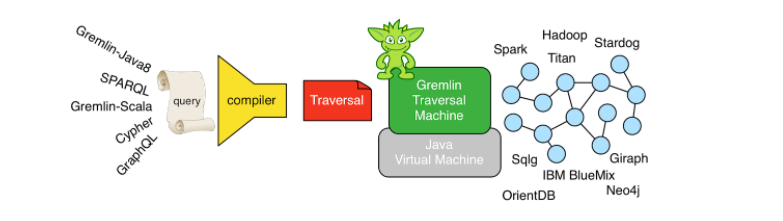
We can use Hadoop echo system

API to build knowledge graph:

Stats Data with Taxonomy:

* This api will focuse on getting data from the stats/report db and taxonomy and create a knowledge graph based on the ontology
* We can use client side language like c#, python and scala
* To get data from rdf we can use sparql or apache thinker pop
* Apache thinker pop is graph computation frame work, it has wrappers around most of the client language, u can run sparql quires also in it also ,
* We can use thinker pop as an abstraction layer for dbs and get the data to system and build the knowledge graph on the data.
* It comes with graph traversal language called gremlin

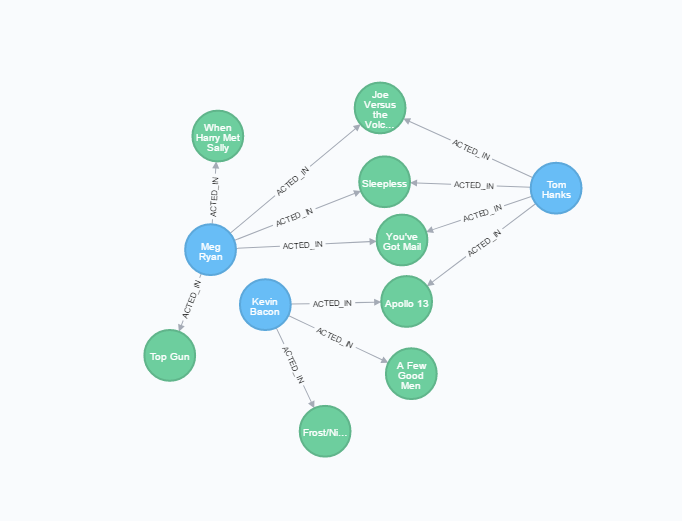




Benefits of gremlin:

Link: https://www.datastax.com/dev/blog/the-benefits-of-the-gremlin-graph-traversal-machine

This is how the knowledge graph will look like



Crawled data with Taxonomy:

* This api will focused on getting data from the Hadoop and taxonomy and create a knowledge graph based on the ontology.
* We can use natural language processing, machine learning or neural network to figure out the best taxonomy and ontology to fit the data for all the text.
* We can have some Scala or python jobs to run on Apache spark to structure the data.
* We can have separate frame work to handle both the structures or have a common frame work to handle it.
* We can have python or Scala to handle to heavy load of data by running on spark.
* Once the relationship is formed we can add it to the knowledge graph

Knowledge Graph:

This will be a central intelligent graph system which contains the relationship between data points

Here each data points will have link to the actual data using id or something to identify the data.

Knowledge graph should be a property graph because we need the graph to be as smart as possible.

Here we need to discuss on multiple dbs and frame work to understand which fits best

We have multiple dbs like Neo4j, cosmos DB, Amazon Neptune, Data Stax, Titan DB, GraphX.

Neo4j (https://neo4j.com/):

Open source graph database.

It’s a complete graph db to the core.

Community is very good and well documentation.

Build in algorithm like Dijkstra, shortest path and so on.

Api and access methods : cypher Query language, restfull http , thinker pop3

Supports distributed graph database ,

GPL v3 license that can be used all the places where you might use MySQL.

But in enterprise edition which may cost around 20k $ (not the exact number but looks like around it, based on talks in video )

Microsoft Azure Cosmos DB:

Globally distributed, horizontally scalable, multi-model database service

Microsoft product over all community is good documentation is good

Api access: Graph API (Gremlin), RESTful HTTP API

Cost: ssd storage: $0.25 GB/month

Reserved RUs/second (per 100 RUs, 400 RUs minimum): $0.008/hour

|  |  |  |
| --- | --- | --- |
| Item | Usage (Month) | Monthly Cost |
| Throughput bill for 3 additional regions - East US, North Europe, and East Asia | 3 \* 10K RU/s \* 24 \* 31 | $1,785.60 |
| Storage bill for container in West US | 1 TB | $256 |
| Storage bill for 3 additional regions - East US, North Europe, and East Asia | 3 \* 1TB | $768 |
| Total |  | $3,404.80 |

Amazon Neptune (https://aws.amazon.com/neptune/)

Amazon Neptune is a fast, reliable, fully-managed graph database service that makes it easy to build and run applications that work with highly connected datasets.

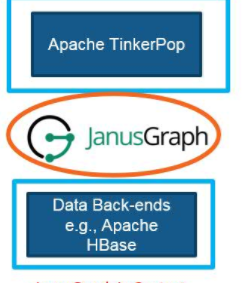
The core of Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds latency.

Amazon Neptune supports popular graph models Property Graph and W3C's RDF, and their respective query languages Apache Tinker Pop Gremlin and SPARQL, allowing you to easily build queries that efficiently navigate highly connected datasets.

Cost:     Storage Rate           $0.10 per GB-month  
            I/O Rate                   $0.20 per 1 million requests

https://aws.amazon.com/neptune/pricing/

Titan Db | janus Graph:



TitanDB was originally developed by a company called Aurelius, was acquired in 2015 by [DataStax](http://www.datastax.com/),

Jaus graph acts a computing layer on top of any data back end

You can use dbs like Hbase , casndra ,

Janus will mimic a graph like structure on top of this data backend

DSE Graph ( product of dataStax)

Link: https://www.datastax.com/products/datastax-enterprise-graph

Uses thinker pop and Goblin as there access layer,

Can be as orm using pythonDSl

GraphX :

GraphX is Apache Spark's API for graphs and graph-parallel computation

When you are working with really large data and need really fast computing we use graphx

U can have multiple data backends with this it .