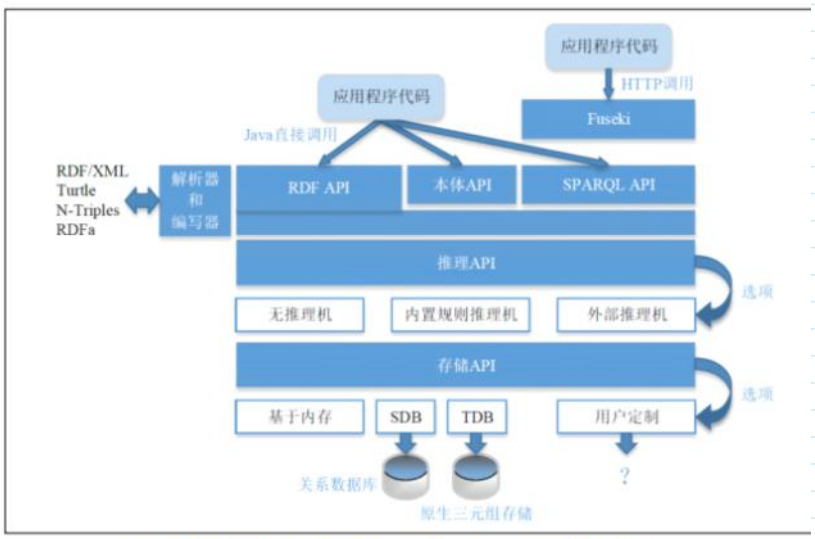
SEU 知识存储

2019年12月1日 13:46

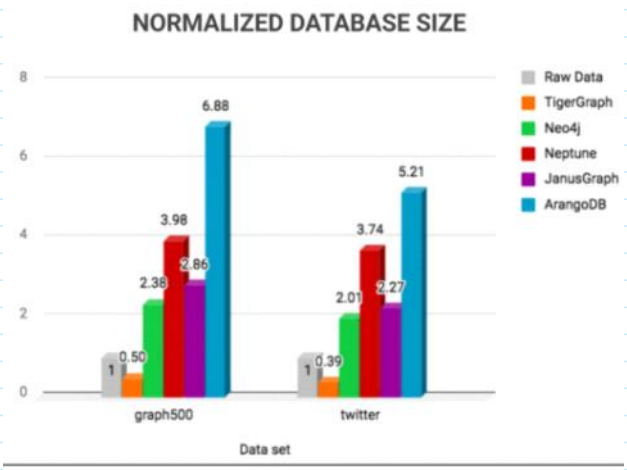
- **知识存储概述**
  - 当我们经过知识抽取得到了用于构建知识图谱的知识，并选择了合适的方法对知识进行建模和表示之后，就要考虑对知识的持久储存
  - 知识图谱以图结构来对知识进行建模和表示，所以常将知识图谱中的知识作为图数据进行存储
  - 试验阶段的小规模知识图谱多使用文件对知识进行存储
  - 在面临大规模知识图谱的查询、修改、推理等需求时，需要考虑使用数据库管理系统（DBMS）对知识进行存储。
  - 常用的数据库 管理系统有：
    - 关系型数据库管理系统（Relational DBMS）
    - 图数据库管理系统（Graph DBMS）
    - RDF 存储系统（RDF Stores）
    - 图数据库管理系统和RDF存储系统使用图数据库模型，可以直接用于知识图谱的存储
    - 关系型数据库通常不会被直接使用于知识存储，但由于其具有成熟的技术体系，使得不少RDF存储系统使用关系型数据库作为底层存储方案，实现对RDF数据的存储。

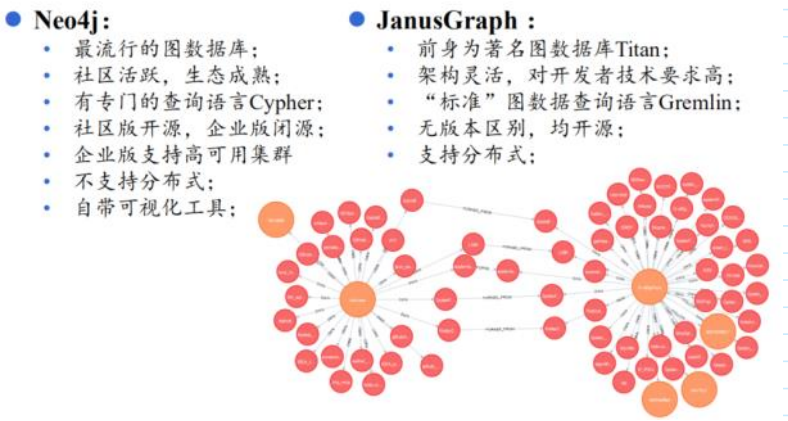


语义网开发框架Apache Jena架构图

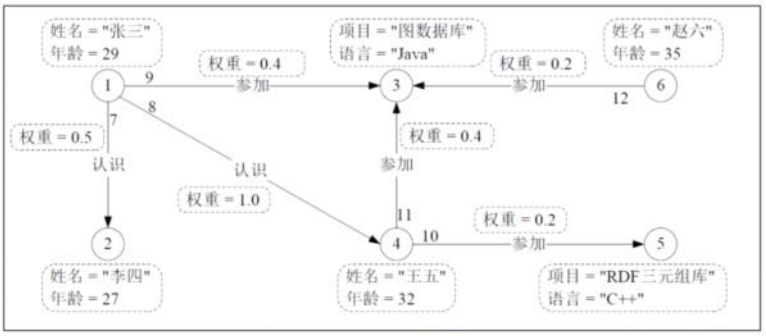
- **图数据库管理系统**
  - 图数据库管理系统：专门用于管理图结构数据 Graph DBMS
  - 属性图是图数据库领域中采用最广的一种数据模型

- 有节点和边构成
- 节点可以有一个或者多个标签（Labels）
- 节点可以有属性（键值对）
- 边有一个类型（Type）和方向
- 边也可以有属性
- 常见的图数据库有：Neo4j，TigerGraph， JanusGraph
    - **性能对比**





- **图数据库查询语言**
    - **Cypher**
        - 声明式（declarative）语言
        - 用户只需要声明“查什么”无需关心“怎么查”
        - Neo4j图数据库专用
    - **Gremlin**
        - 过程式（procedural）语言
        - 图遍历语言，用户需致命具体导航步骤
        - 业界标准，Neo4j除外的机会所有图数据库均支持



Gremlin: *g.v(1).out('认识').filter{it.年龄>30}.out('参加').项目*

查询结果:
图数据库
RDF三元组库

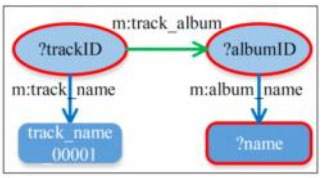
Cypher:
```
MATCH (n)-[:认识]->(p), (p)-[:参加]->(pr)
WHERE n.id==1, p.年龄 > 30
RETURN pr.项目
```

- **RDF存储系统**
  - RDF资源来描述语义网络中的资源并形式化的表示出资源之间隐含的语义关系。
    为此，语义网研究领域发展出专门用于存储RDF数据的RDF存储系统。
  - RDF用三元组的结构来描述资源（主语subject，谓词predicate，宾语object）
  - RDF标准，每个三元组成为陈述statement，多个语句的集合成为描述describtion
  - RDF三元组的集合可看做图数据的一种表示，所以RDF存储系统可以认为是一种严格遵守**语义网络标准**的图数据库
  - **Virtuoso**
    - 多模型混合（Multi_model）数据库管理系统
    - 支持关系数据、属性图（property graphs）数据、RDF数据、XML数据和文件型数据的统一管理
    - 包括DBpedia在内的很多开放只是图谱选择其作为后台存储系统
  - **RDF查询语言SPARQL**
    - 全程为：SPARQL Protocol And RDF Query Language
    - 声明式（declarative）语言
    - 由W3C指定的RDF标准查询语言
    - 被认为是语义网络技术的关键
    - **所有RDF存储系统都以SPARQL为查询语言**
    - **E.G.**

SPARQL查询实例:

查询名为“track_name_00001”的歌曲对应的专辑名。

```
PREFIX m: <http://kg.course/music/>
SELECT ?name
WHERE {
        ?trackID m:track_name "track_name_00001"
        ?trackID m:track_album ?albumID
        ?albumID m:album_name ?name
}
```



- **基于关系型数据库的存储方案**
  - 在关系型数据库上设计合适的存储方案，同样能实现对图结构数据的存储
  - 关系型数据库在数据管理方面高效，使得一些RDF三元组库是基于关系型数据库管理系统而实现的
  - 涉及图的全局操作时，图数据库和三元组数据库的性能往往不如传统的关系型数据库
  - 三元组表：
    - 将图数据用RDF三元组表示

- 每个三元组作为表中的一行记录
- 多跳查询时会产生自连接（Self-join）操作

主语	谓语	宾语
Charles_Flint	born	1850
Charles_Flint	died	1934
Charles_Flint	founder	IBM
Larry_Page	born	1973
Larry_Page	founder	Google
...	...	...

```
SELECT t1.主语
FROM t AS t1, t AS t2, t AS t3
WHERE
t1.主语 = t2.主语 AND t2.主语 = t3.主语
AND t1.谓语 = 'born' AND t1.宾语 = '1850'
  AND t2.谓语 = 'died' AND t2.宾语 = '1934'
  AND t3.谓语 = 'founder'
```

查询某个生于1850死于1934且创建过公司的人

- 水平表：
  - 每行存储一个主语对应的所有谓语和宾语
  - 只适用于为此数量较少的知识图谱
  - 对于一个主语，可能只在极少的列上有值，导致稀疏
  - 无法存储多值属性或一对多联系

主语	born	died	founder	board	...	employees	headquarters
Charles_Flint	1850	1934	IBM		...		
Larry_Page	1973		Google	Google	...		
Android							
Google					...	54,604	Mountain_View

- 属性表：
  - 相当于对水平表的划分
  - 将不同类型的实体存入不同的表中
  - 适用于实体种类较少的情况
  - 仍然不能存储多值属性或一对多联系
  - 是一种常见的存储方案

person

主语	born	died	founder	board	home
Charles_Flint	1850	1934	IBM		
Larry_Page	1973		Google	Google	Palo_Alto

os

主语	developer	version	kernel	preceded
Android	Google	4.1	Linux	4.0

company

主语	industry	employees	headquarters
Google	Software, Internet	54,604	Mountain_View
Larry_Page	Software, Hardware, Services	433,362	Armonk

- 垂直划分
  - 相当于对三元组表按谓词进行划分
  - 为每种谓词创建一张两列的表
  - 解决了稀疏性及多值属性的问题
  - 设计多个谓词的查询将导致多表连接的操作
- 无论使用哪种方案，图数据的大多数多跳查询都会导致关系数据库中的连接（join）操作。所以在进行图遍历时关系数据库会产生比较大的开销，此时图数据库的性能优于关系型数据库
- 关系型数据库的成熟的技术架构，使得其对数据的全局操作（计数、求和）有着更为高效的性能，而这方面恰好是图数据库的不足之处

- **总结**

	图数据库管理系统	RDF存储系统	关系型数据库
数据模型	属性图	RDF三元组	关系数据模型

	图数据库管理系统	RDF存储系统	关系型数据库
数据模型	属性图	RDF三元组	关系数据模型
查询语言	Cypher、Gremlin	SPARQL	SQL
应用场景	多为工业界场景	多为学术界场景	学术界和工业界均有应用
其他特点	图遍历效率较高； 图的全局操作效率较低；	有标准的推理引擎； 易于发布数据；	多跳查询会产生自连接操作，影响查询效率；

- 选择知识存储方式的时候，需要根据具体的数据规模以及应用场景选用适合的存储方式
- 绝大多数的实际应用场景不会涉及到多跳查询（三跳以上）这时候选择传统的关系型数据库，设计合适的Schema也可以满足应用需求
- 当应用涉及多跳查询、计算最短路径、推理分析等需求时，图数据库和RDF数据库的优势才得以体现