图数据库

Graph Database

——180616 班 18373528 杨凌华

一、产生背景

“As we drilled deeper into the persistence layer of our enterprise content management application, we realized that our software was managing not just a lot of individual, isolated, and discrete data items, but also the connections between them. And while we could easily fit the discrete data in relational tables, the connected data was more challenging to store and tremendously slow to query.”[1]

—Emil Eifrem

Cofounder of Neo4j and CEO of Neo Technology

从历史上来看，大多数的企业级 web 应用的运行都是建立在关系数据库的基础之上，但是在过去近二十年间，人们一直在经历着一个同样的问题，那就是数据量的不断增大，数据在内容乃至结构上的变化之频繁，之迅速，已经逐渐超过了传统的关系型数据库管理系统 RDBMS 的处理能力，为了迎接这样的挑战，NoSQL 逐渐走出实验室，走进实际应用当中。

在数据量急剧上升的大背景下，数据体积(Volume)成为了数据库存储结构设计的最主要驱动和制约因素。体量大的数据集存储在关系数据库中会显得格外笨重，比较典型的是，数据库查询操作的执行时间，会随着表的尺寸增大以及表之间 join 连接操作的增加而急剧上升，这就是所谓的”join pain”效应。问题并不出在数据库本身，而是在于其底层的数据模型的构建方式，为了解决这一问题，NoSQL 大家族中便产生了图数据库这一针对于大型数据库更高效的数据库模型构建方式，它不仅在处理大数据量上性能远高于关系数据库，而且在关系表达上更加贴近于自然关系描述。

除此之外，在数据不可能是静态的，它是一个时刻处于变化过程中的不稳定的动态的集合，这种不稳定，不仅表现在数据内容的多变，更表现在包括数据内容在内的数据整体结构的改变，其原因有两点：其一，数据库处理事务的不断改变，引起数据结构的改变；其二，对于同一个事务，其数据的需求也是动态的，在不同时刻所需要的的数据并不一样，因此没有必要一次调动所有数据，而应该动态的调动当前所需要的一部分数据即可。对于传统关系数据库来说，高写入负载会导致很高的处理成本，高架构波动性也将导致很高的运营成本，相比之下，图数据库在处理动态数据时具有显著的优势。

二、基本原理

图数据库(GDB)是一种基于图论的数据库，它通过将数据转化成结点，将关系转化成连接在结点与结点之间的边，来对数据用直观自然的视角来进行描述[2]。与关系型数据库相比，图数据库更直观、更简单、语义更丰富，搜索效率更高，且非常适合大数据的存储和检索[4]，可广泛应用于诸如社交关系、组织架构、交通信息、网络拓扑等场景。

图数据库具备图形结构数据的原生存储和遍历能力，由于图形数据结构关系变化的多样性，图数据库适合对数据结构较为复杂的关联关系、动态关系变化较快的海量数据进行存储 和管理，可以对数据关联关系进行快速匹配、遍历和查找。

图数据库的组成部分如下：

结点(Nodes)：代表实体或实例，比如人、事务、账户或者其他需要被追踪的事物。它们基本等价于关系数据库中的记录、关系或者实体元组。

边(Edges)：也称为图线、关系，是连接在结点与结点之间的连线，代表结点与结点之间对应的实体之间的关系。当考察结点、属性、边之间的连接和相互关联时，就会产生相关的语义信息。边可以分为有向边和无向边，在无向图中，结点和结点之间的联系只有一种含义，而在有向图中，结点与结点之间的联系会因为边指向的不同而各有含义。边是图数据库中一种典型的抽象，这种抽象是关系数据库中所不具备的。

属性(Properties)：是指关联在结点、边上的信息，描述了一个结点所代表的实体的相关属性信息，以及结点与结点之间联系的信息，它采用的是键值对(key-value pairs)的方式进行存储，具体内容依据数据库实际的需求而填充。

标签(Labels)：标签用于将结点分组，一个结点可以具有多个标签，对标签进行索引可以加速在图中查找结点。

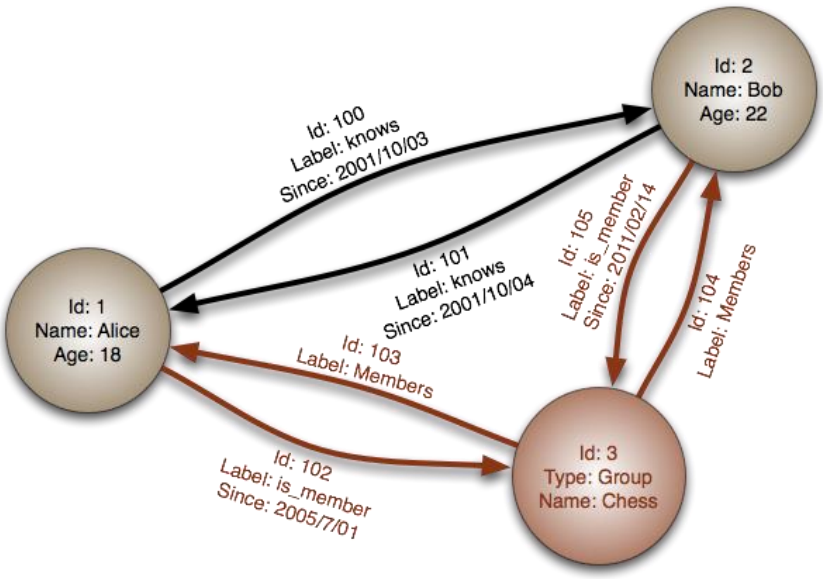

图 1 一个带有结点、属性和边的图示例[2]

三、特点

图数据库更适用于相互之间高度关联的数据结构，可以通过更贴近现实关系的方式进行图数据的建模，它在处理多数据关系建模、数据关系动态扩展、实时遍历数据关系等方面具有诸多优点：

（1）基于图的遍历方式访问数据(Graph-traverse)：

图数据库具有强大的图查询方式，比如能很轻易的计算出两个结点之间的最短路径、直径计算、群体检测等。

（2）灵活性、扩展性强(Flexibility)：

图数据库能够使用户在不对原有功能产生破坏的情况下，对原有的图插入新的数据，也就是对于数据库的构建和维护者来说，不必提前预测规划需求，不需要为将来数据库的扩展预留接口，大大降低了数据库的维护难度。同时，在数据集增大时，它的性能趋向于保持不变。

（3）高性能的数据定位(High Performance)：

相比于关系数据库通过查询关系表、将两实体表进行 join 连接这样的操作来说，图数据库采用**无索引邻接**(Index-free Adjacency)的机制对数据进行定位，它让每一个结点都保留有直接指向其他邻接点所在物理存储的指针，从而能够进行快速的从结点到结点之间的索引，这样一来就不需要像关系数据库那样通过访问其他形式的数据结构来获取两结点之间的联系。同时通过引入高速缓存 cache 机制，一旦一个结点被访问过，那么在一段时间内，其访问的性能效率会进一步提升。当然其局限性在于，如果访问不是采用图遍历的方式，那么其访问效率就会有所降低。

关系深度	MySQL 执行时间	Neo4j 执行时间
2	0.016 秒	0.01 秒
3	30.267 秒	0.168 秒
4	1543.505 秒	1.359 秒
5	未完成	2.132 秒

表 1 关系型数据库和图数据库深度查询性能对比[3]

（4）敏捷性(Agility)：

得益于图数据库的无模式性(schema-free)和应用程序接口以及查询语言的可测试性(testable)，可以帮助开发者以一种可控的方式进行应用的演进，图数据库不需要关系数据库那样的面向模式(schema-oriented)的数据治理机制，它采用更

加可视化、可操作性更强的治理方式，这种方式能很好的适应当下敏捷开发、测试驱动开发的软件开发实践。

四、Neo4j 介绍

4.1 基本情况

Neo4j 是由 Neo Technology 开发的当前较为主流和先进的原生图数据库之一，提供原生的图数据存储、检索和处理。

1) 它支持 ACID、集群、备份和故障转移，具有较高的可用性和稳定性；

2) 它具备非常好的直观性，通过图形化界面表示结点和关系；

3) 它具备较高的可扩展性，能够承载上亿的结点、关系和属性，通过 REST 接口或者面向对象的 JAVA API 进行访问。

节点	2^{35}（~ 340 亿）
关系	2^{35}（~ 340 亿）
属性	2^{36} to 2^{38}, 取决于属性类型 （最大 ~ 2740 亿，通常至少 ~ 680 亿）
标签	2^{15}（~ 32000）

表 2 Neo4j 各部分数据量支撑情况[3]

4.2 存储结构

Neo4j 图数据库的不同部分(结点、关系、属性、标签)分别保存在不同的存储文件(Store File)中，它对各个存储文件进行了专门的设计和优化以提升存储和访问效率，其运行时引擎可以对这些文件格式进行高效的查找和遍历。

存储文件名	记录容量	内容
neostore.nodestore.db	14 字节	节点
neostore.relationshipstore.db	33 字节	关系
neostore.propertystore.db	41 字节	属性
neostore.propertystore.db.arraysneostore	120+8 字节	数组属性的值
propertystore.db.strings	120+8 字节	字符串属性的值

表 3 Neo4j 主存储文件及相关的属性[3]

Neo4j 同样采用无索引邻接(Index-Free Adjacency)机制存储数据，这样当执行遍历时就能够直接跟随指针连接结点和关系，相对于关系型数据库的访问速度更

快。

4.3 查询语言

Neo4j 有基于自身优化的一套图形搜索算法，采用 Cypher 这种简洁且富有表现力的图数据库查询语言来以精确的方式程序化地描述图结构，并对图形数据进行增删改查(CRUD)操作。具体的在搜索功能中，它由一下四部分组成[3]：

1) start：在图中指定一个或多个起始结点，可以通过索引查找获得，也可以通过结点的编号直接获得；

2) match：图形的匹配模式，也是实例化的需求部分；

3) where：提供过滤模式匹配结果的条件；

4) return：用来指明在已经匹配查询的数据中，哪些结点、关系和属性是需要返回给客户端的。

五、应用场景

图数据库在社交关系、组织架构、交通信息、网络拓扑等很多场景都有着广泛的应用，下面以图书馆[3]的各种业务来举例：

1) 图书借阅关系建模：

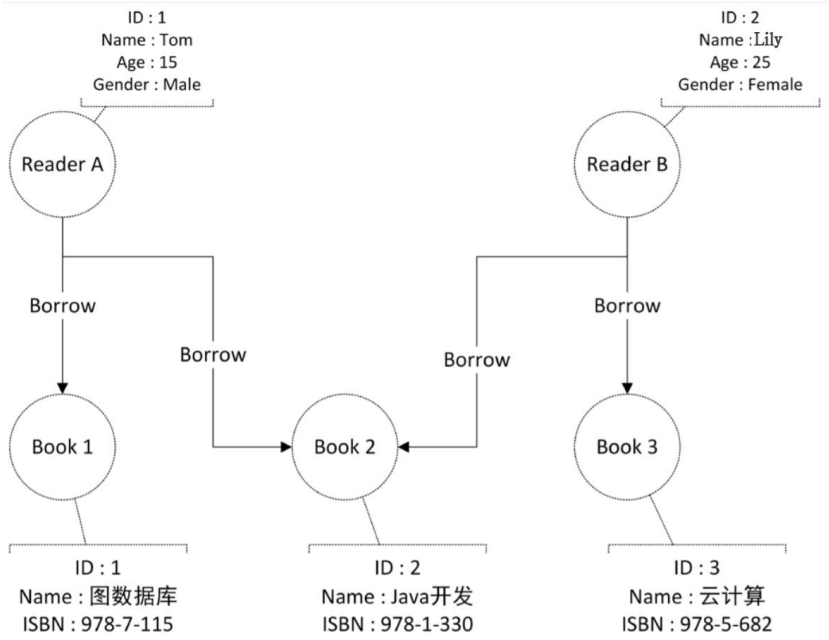

图 2 图书馆借阅关系模型图

```
create (n:Reader {id:1, name:'Tom', age:'15', gender:'male'}) return n;
match (r:Reader),(b:Book) where r.id=1 and b.id=1 create
(r)-[borrow:Borrow]->(b) return r,borrow,b;
```

图 3 图书馆借阅关系建模语句

2) 读者喜好和评论功能建模：

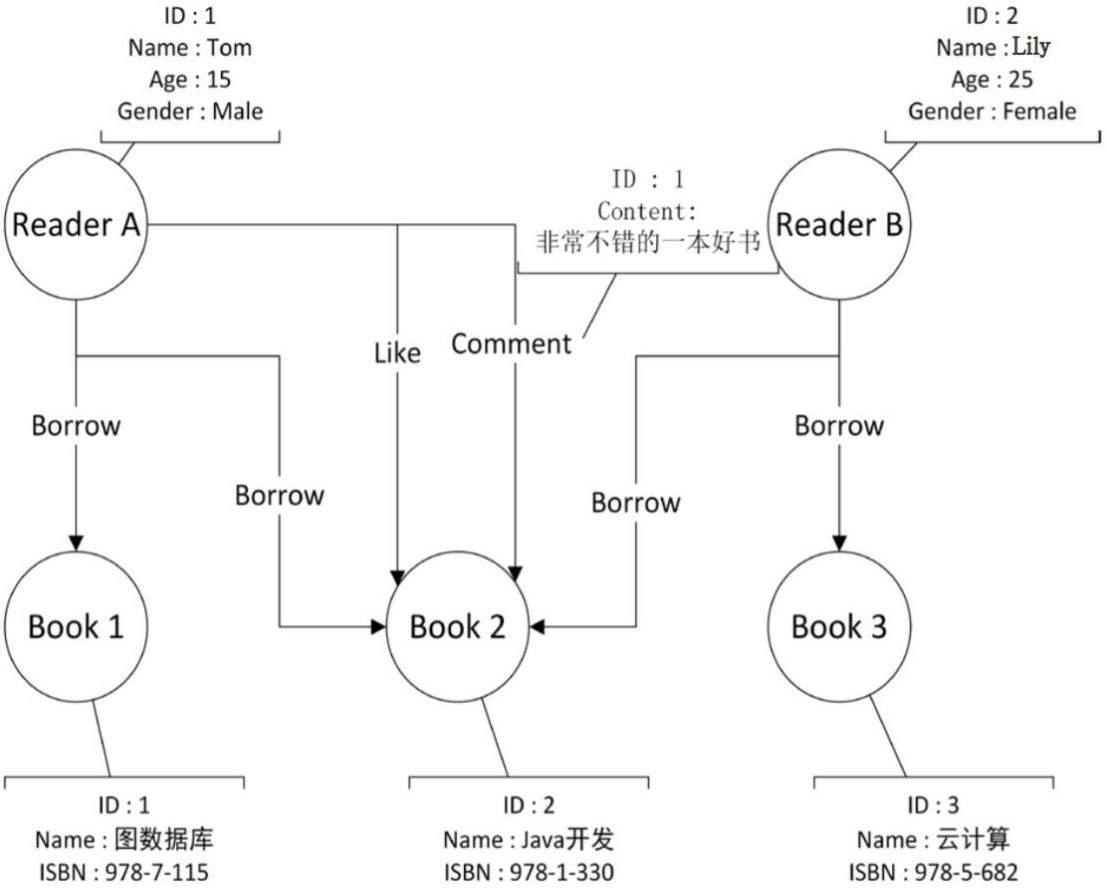

图 4 读者喜好与评论功能模型图

```
match (r:Reader),(b:Book) where r.id=1 and b.id=2 create (r)-[like:Like]->(b) return r,like,b
match (r:Reader),(b:Book) where r.id=1 and b.id=2 create (r)-[comment:Comment {ID:1, content:'非常不错的一本好书'}]->(b) return r,comment,b
```

图 5 读者喜好与评论功能建模语句

3) 读者社交关系建模：

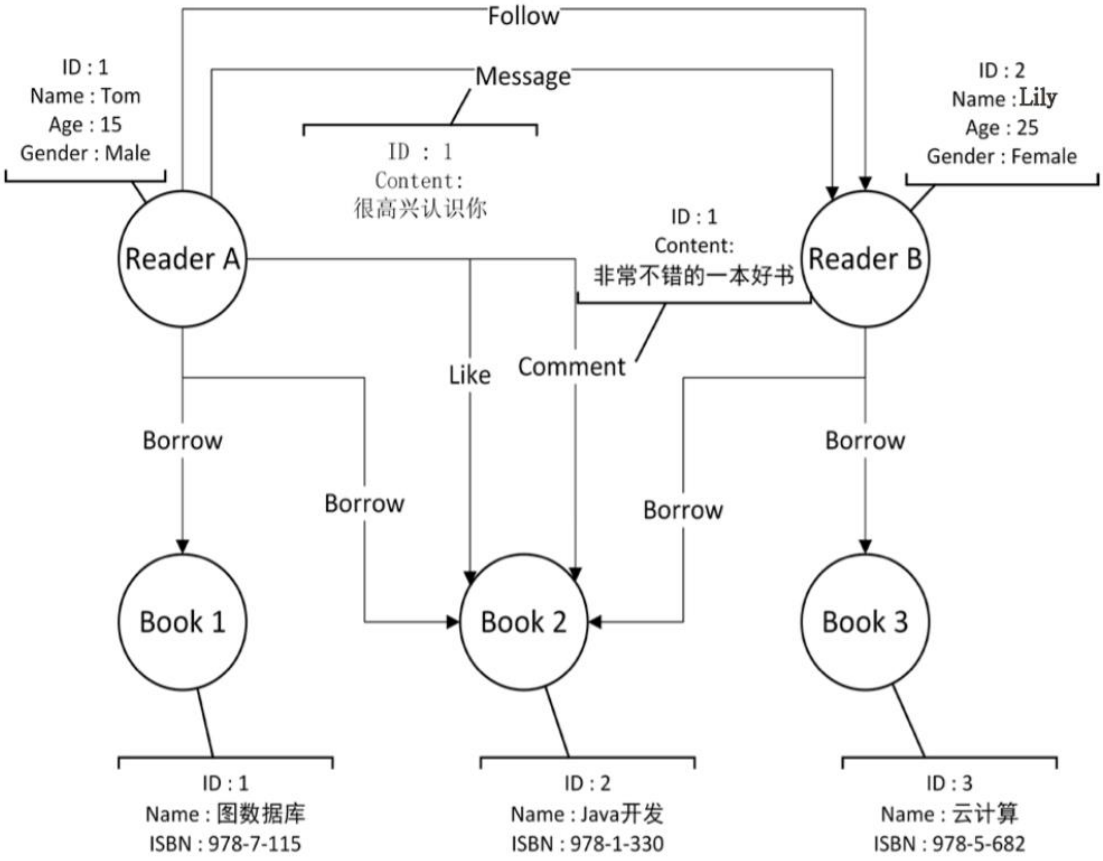

图 6 读者社交关系模型图

```
match (r1:Reader), (r2:Reader) where r1.id=1 and r2.id=2 create (r1)-[follow:Follow]->(r2) return r1,follow,r2
match (r1:Reader), (r2:Reader) where r1.id=1 and r2.id=2 create (r1)-[message:Message {ID:1, content:'很高兴认识你'}]->(r2) return r1,message,r2
```

图 7 读者社交关系建模语句

4) 读者图书推荐系统

要达到一个真正意义上完善的推荐系统，图书馆需要针对每一位读者进行个性化需求分析从而进行智能化推荐。可以通过引入图数据库，以阅读内容为联系结点构建群体阅读网状模型，基于与图书的关联对读者群体进行分类，将每一位读者作为一个结点，同时也将其阅读的每一本书作为结点，二者之间建立阅读关系，读者与读者之间也能通过阅读同一本书而产生联系，通过这样一种错综复杂的阅读关系，不同读者之间便产生了社交联系，进而构建了一张庞大的社交关系网，系统基于相似群体阅读内容的特性，为读者推荐适合其需要的个性化的阅读内容。

参考文献

[1] Robinson I, Webber J, Eifrem E. Graph Databases[M]. Cambridge: O'Reilly Media, 2015.

[2] Wikipedia contributors. (2020, December 18). Graph database. In Wikipedia, The Free Encyclopedia. Retrieved 06:43, December 23, 2020, from https://en.wikipedia.org/w/index.php?title=Graph_database&oldid=995023418

[3] 李金阳.图数据库在图书馆的应用研究[J].图书馆,2020(11):109-115.

[4] 邱胜海,王云霞,樊树海,贾晓林.云环境下图数据库建模技术及其应用研究[J].计算机应用研究,2016,33(03):794-797.