英文数据库有哪些（常见的数据库管理系统）

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/9b8cab8f9923d8f9baea1844634e2990.html

在计算机科学中，**图数据库**（英语：graph database，**gdb**）是一个使用图结构进行语义查询的数据库，它使用节点、边和属性来表示和存储数据。该系统的关键概念是**图**，它直接将存储中的数据项，与数据**节点**和节点间表示关系的**边**的集合相关联。这些关系允许直接将存储区中的数据链接在一起，并且在许多情况下，可以通过一个操作进行检索。图数据库将数据之间的关系作为优先级。查询图数据库中的关系很快，因为它们永久存储在数据库本身中。可以使用图数据库直观地显示关系，使其对于高度互连的数据非常有用。

图数据库是一种非关系型数据库，以解决现有关系数据库的局限性。图模型明确地列出了数据节点之间的依赖关系，而关系模型和其他 nosql 数据库模型则通过隐式连接来链接数据。图数据库从设计上，就是可以简单快速地检索难以在关系系统中建模的复杂层次结构的。图数据库与 20 世纪 70 年代的网络模型数据库相似，它们都表示一般的图，但是网络模型数据库在较低的抽象层次上运行，并且不能轻松遍历一系列边。

图数据库的底层存储机制可能各有不同。有些依赖于关系引擎并将图数据“存储”到表中（虽然表是一个逻辑元素，但是这种方法在图数据库、图数据库管理系统和实际存储数据的物理设备之间施加了另一层抽象）。另一些则使用键值存储或面向文档的数据库进行存储，使它们具有固有的 nosql 结构。大多数基于非关系存储引擎的图数据库还添加了**标记**或**属性**的概念，这些标记或属性本质上是具有指向另一个文档的指针的关系。这样就可以对数据元素进行分类，以便于集中检索。

从图数据库中检索数据需要 sql 之外的查询语言，sql是为了处理关系系统中的数据而设计的，因此无法“优雅地”处理遍历图。截至 2017 年，没有一个像 sql 那样通用的图查询语言，通常都是仅限与一个产品的。不过，已经有一些标准化的工作，使得 gremlin、sparql 和 cypher 成为了多供应商查询语言。除了具有查询语言接口外，还可以通过应用程序接口（api）访问一些图数据库。

> 图数据库与图计算引擎不同。图数据库是转换关系 oltp 数据库的技术。而图计算引擎在 olap 中用于批量分析。由于主要技术公司在使用专有图数据库方面的成功以及开源图数据库的引入，图数据库在 2000 年代引起了相当大的关注。

上面部分引用了维基百科对图数据库的词条来讲解何为图数据库，而本文整理周公解梦梦见死人于图数据库 nebula graph
交流群中对图数据库的零碎知识，作为对图数据库知识的补充。本文分为小知识及 q&a 两部分。

本文主目录

小知识

学习图数据库的起手式——了解图数据库兴起的契机。

图数据库兴起的契机 – @阿秾

2010 年前后，对于社交媒体网络研究的兴起带动了图计算的大规模应用。

2000 年前后热门的是 信息检索 和 分析，主要是 google 的带动，以及 amazon 的 e-commerce 所用的协同过滤推荐，当时 collaborative filtering也被认为是 information retrieval 的一个细分领域，包括 google 的 pagerank 也是在信息检索领域研究较多。后来才是 twitter，facebook 的崛起带动了网络科学 network science的研究。

图理论和图算法不是新科学，很早就有，只是最近 20 年大数据，网络零售和社交网络的发展，big data、social networks、e-commerce 、web 2.0让图计算有了新的用武之地，而且硬件计算力的提高和分布式计算日益成熟的支持也使图计算在高效处理海量数据成为可能。

学习完图数据库发展的契机，我们来学习下图数据库存储方式和一种图数据库存储层的设计探讨。

图数据库存储方式 —— 基于内存存储 vs 基于分布式 kv 存储 – @bruceleexiaokan

bruceleexiaokan：基于内存的图数据库有其优势，特别对于 大规模深度遍历以及基于之上的 graph model 计算，这在大规模并行处理（ mpp ）是有较强优势，其访问语言更像是编程语言而并非图遍历。

sherman：各种存储各有优缺点，各有擅长的应用场景，所以离开了场景和需求，很难对比不同的解决方案。

bruceleexiaokan：基于分布式 kv 之上的图数据库，对于大规模深度遍历和计算，对于graph model 的支持，有其缺陷。图数据库需要有分类，我们需要明白讨论的是哪一种。

实时在线图数据库，线下图数据库，大规模数学分析用图数据库。

如果讲到第 3 种，图结构基于内存的方案有优势。第 1 和 2 种大规模图数据库主要也就是基于 kv+ 索引

一种图数据库存储层的设计探讨 – @bruceleexiaokan

无中心化的存储集群，一般单个集群还是有一定的大小限制，不宜过大。存储层的抽象在于，数据集（图的话就是不同的点和边）到存储集群的逻辑映射对用户透明，用户可用性要求高的场景需要考虑双集群互为灾备。单集群的数据平衡是集群内部的事，集群和集群间的数据平衡是需要设计的，其中线下到线上的数据传输通道尤其重要。
设计原则：

不要使得单集群过大；本地互为备份集群支持读 active-
active；利用线下到线上数据传输通道做好数据集群间迁移、backfill、recovery，batch update 等等工作：数据访问有抽象，使得集群的运维对于用户访问透明；做好集群间的跨数据中心数据复制；到达即使逐步投资也能线性扩展的设计；

学习完存储和设计的小知识，来对比下图数据库图结构的可视化和 gis 数据的可视化。

图结构的可视化与 gis 数据的可视化 – @space

关于图结构可视化与 gis 数据的可视化本质上有比较大的差异：

gis 是 hierarchical + 瓦片式贴片展示的，而图结构本身是 flat 的，只能一次性将所有 touch 到的数据全部展示出来。但是 gis 的做法可以给我们启示，结合具体的业务场景，能否也做一个层级抽样，但是图抽样的问题是：如何在抽样的同时，尽量 保留子图的连通性（否则可能 high level 的层显示的都是孤立的点，只有最后最细粒度的层才会显示所有数据）。
一些粗浅的想法：可以结合图计算的技术，先算连通子图，然后在连通子图内部算 pagerank，按照 pagerank 大小划分成不同的区间，相当于按照 pagerank 值做 hierarchical 分层，在层次切换时，为了保证图的连通性，除了显示下一个层次的顶点（pagerank 值在下一个区间）之外，还需要显示这 2
个层次抽样出来的顶点的边（这相当于一个子图内部的连通路径的检索，如果能做 aggreate 更好，如果这些边很多，是否可以按照 edgetype
aggregate，先显示统计值，如果用户有兴趣再展开——即图数据库返回 aggregation 值，前端生成 ” 虚拟 ” 的边，随着进一步展开，这些 ” 虚拟的边 ” 会被实际明细边取代）。

上述 trick 只是为了解决图数据像 gis 一样平滑展示的问题，缺点也比较明显，hierarchical 抽样代价高。

另外，图数据的展示问题，不是一个独立的前端技术问题，还涉及到后端图数据库如下 feature 的支持：

degree 统计按照 edgetype 进行 aggregationquery 时遇到超级顶点做截断，并返回截断信息给 client

内置一些 ap 性算法，如 pagerank、lpa、环探测等。

图数据可视化，还需要考虑：前端数据承载量是有限的，cs 类型的可视化工具还好点，bs 类型的可视化工具，浏览器承载的量就更少。如何在业务上将 touch 到的数据量限制在一定范围内是应用是要考虑的。此外，由于顶点和边的 name 和其他 tag 信息，一般在可视化的时候不会一次性都显示在图上，首次绘制可仅向图数据库请求 name，后续 tag 的 properties 在用户感兴趣的时候（点击/hover）时再次请求。

布局问题：目前常见的无非是力导引、圆形、树形、网格型，这些都是无任何业务语义的布局，如树形布局，哪些应该作为顶层节点，哪些是下一级节点，如果仅仅通过边的有向性，单个 edgetype 显示还好，多个 edgetype 混合在一棵树上显示的时候会破坏掉单个 edgetype 树的结构，必须引入业务规则来限制不同布局下的问题

q&a 提问回答

由于 q&a 整理于 nebula graph 交流群，有多人参与讨论，所以以下问题回复中会有群友昵称出现，不做 nebula graph 官方成员和群友身份区分，仅交流图数据微克和毫克怎么换算库技术~如果你对下列问题有不同的看法欢迎本文评论区交流(≧▽≦)，加入图数据库交流群请加 wechat：nebulagraphbot

图数据库计算存储分离设计及该设计模式的考量原因

提问：计算存储分离的话，数据迁移，请问下大佬们，网络带宽会是瓶颈吗？nebula 怎么解决的呀？

恒子：现在都万兆网卡了，一般机房内很难把带宽打满的，通常 io 会先是瓶颈。

波娃子：如果是地理分布式的图数据库，带宽是要考虑的性能限制因素。

sherman：是的，现在比较流行的做法是两地三中心或者三地五中心。分布式图数据库，既有图的部分，也必然会涉及到分布系统的部分

bruceleexiaokan：由于大规模在线图数据库都设计成计算和存储分离，数据存储的设计是尤为重要。就金融 risk 而言，逻辑上其实就是一张大图，有上百 tb 的数据量，可线性扩展的存储层设计是图数据库的关键

提问：为什么都设计成计算存储分离的模式，有什么重要的考量吗

bruceleexiaokan：对于 risk 而言，在线是 inference 为主，大部分场景是为了 feature 计算，基本在 2-3 跳以内的图遍历，都很简单，但是对于性能和可用性的要求很高，所以在线图数据库存储分离很合理。但针对数据分析的图数据库，其设计会不一样，更需要的是图的深度遍历能力，因此存储分离应该是个问题，但如何支持大规模的图，如何 scale up 应该是关键，而不是 scale out。

天师：存储计算分离大多是适应云计算架构：存储层买空间，计算层买弹性虚机。

吴敏：长期看，计算、存储和网络几个硬件模块发展的速度是不太一样的，并不都是摩尔定理的速度，分离能更合适长期硬件演进

sherman：我觉得存储计算分离的一个很大的好处是存储集群和计算集群可以独立扩缩容，可以通过对不同集群容量的调整，最终达到能够满足业务需求的最佳搭配。

怎么理解图数据库顶点和标签

提问：怎么理解 vertex 和 tag 之间的关系，schema 里面有没有 vertex 的概念？一个顶点 id 可以对应多个 tag 是这个意思吗？

sherman：解释一下 vertex，tag，edge 以及他们之间的关系：

vertex 是一个顶点，用一个 64 位的 id 来标识，一个 vertex 可以被打上多个 tag（标签），每个 tag 定义了一组属性。

举个例子，我们可以有 person 和 developer 这两个 tag，person 这个 tag 里定义了姓名、电话、住址等等信息，developer 这个 tag 里可能定义了熟悉的编程语言、工作年限、github 账号等等信息。一个 vertex 可以被打上 person 这个 tag，这就表示这个 vertex 代表了一个 person，同时也包含了 person 里的属性。另一个 vertex 可能被同时打上了 person 和 developer 这两个 tag，那就表示这个 vertex 不仅是一个 person，还是一个 developer。

sherman：vertex 和 vertex 之间可以用 edge 相连，每一条 edge 都会有类型，比如是好友关系。每个 edge type 也可以定义一组属性。edge 一般用来表示一种关系，或者一个动作。比如，peraon a 给 person b 转了一笔钱，那 a 和 b 之间就会有一条 transfer 类型的边，transfer 这个边类型（edge type）可以定义一组属性，比如转账金额，转账时间等等。

sherman：任何两个 vertex 之间可以有多种类型的边，也可以有多条同种类型的边，比如转账，两个 person 之间可以有多笔转账，那么每笔转账就是一条边。

提问：对于例子有一个小小疑问，这里的 tag 可以理解为本体 ontology 吗？

sherman：按我的理解，ontology 应该是整张知识图谱，也就是说包含 vertex 和 edge。在 nebula 里，vertex 本身不含内容（也就是说没有属性），内容是存放在 tag 里的，这里“内容”指的是 ontology 里的concept，“边”就是 ontology 里的 relationship。

提问：追加个问题: 多个标签是否支持层级关系，比如组织架构什么的？谢谢？

在 nebula 里，可以定义标签之间的依赖关系，比如上面的例子里，developer 依赖 person。

nebula 如何处理 id 冲突问题

提问：如果要构建一个网络，用户，商家，公众号，文章，这些 id 会重复冲突的。根据现在 vertex id就可以唯一指代点的原则，原有的 id
不能直接使用，有什么办法构建出这个网络吗？还是把 id 作为tag属性，然后建索引。

吴敏：类型和原始 id 拼在一起 hash，作为 vid，然后把原始 id 作为一个 property。

sherman：由于业务千变万化，所以当初我们决定把如何产生 vid 交个业务来决定。vid 是一个 64 位整数，在你的 case 里，如果 id 不足 64 位，那就可以用 2-4 bit
来表示不同的类型，这样就把原来可能冲突的 id 分到了不同的空间。如果原来的 id 已经是 64 bit 的了，那可以像@吴敏岁月是朵双生花 说的那样做 hash，把真实 id 保存在属性里

nebula graph 和 tiger graph 的区别

提问：大佬们，我们想了解下 nebula graph 和 tiger graph有关系，二者有什么区别么

sherman：简单的讲，tiger graph 不是真正意义上的对等分布式，它是有中心节点的分布式，它分布存储的是点和边上的属性，但是整张图的关系必须保存在一台机器上。同时在运行的时候，整张图必须加载到内存里，这就限制了它能处理的图的规模。而一个产品的架构一旦建立之后，要改动不是一件容易的事情，基本相当于重做。

j.guardian：简单理解的话，tiger graph 为了性能牺牲了图规模的处理能力，而 nebula 解决的图规模的能力，但是相对会稍微牺牲一些性能。

sherman：也不完全是，当然这是我之前对 tiger graph 的了解。

图数据库 0 标签的意义

提问：我看我们的文档里写着“一个顶点必须至少有一个类型的标签”，但是我注意到 neo4j 是支持 0 个标签的，请问没有标签的节点在查询时跟普通标签用法一样么，为什么要支持 0 个标签呢？这样做有什么意义呢？

sherman：多数的图计算性能评测的数据集（如 graph500、twitter）都是 0 标签，也就是无属性过滤条件。这样能看出一个图引擎的最核心的性能。通过标签过滤在大多数情况下对图进行动态剪枝，时耗进而儿会缩短。

大家怎么看「图数据库要有索引」这个问题？

提问：大家怎么看「图数据库要有索引」这关于清明节的诗歌个问题？

bruceleexiaokan：最终这是一个设计上的trade off问题，不同数据分布和不同访问需求对于不同的设计方案，性能肯定是不同的。最好的方案是设计存储访问抽象，保留设计和实现灵活性，针对不同场景可以有不同优化。

相邻边的索引和节点 inline 存储本是一种优化，可以减少物理磁盘 block read 数量，和节点一块读和写。但到了一些特殊场景：

如果更新非常频繁，会造成写放大问题单节点边出入度异常高，但访问只遍历前几个。其性能反而会变差，属性索引是另一个问题

sherman：@bruceleexiaokan 完全同意，索引的使用要看场景，过度使用索引会得不偿失。

提问：nebula 是对临接点有索引的 对吧

sherman：对属性有索引

在知识图谱场景下计算、存储及副本一致性问题

提问：我们知识图谱业务场景，查节点间的路径，请问下实时计算结果的效率怎么样呀？还是说比较推荐离线计算？nebula 是存储计算分离的是吧？

sherman：说一下个人理解，我觉得知识图谱的场景一般是需要在线查询的，因为不知道会有怎么样的查询问题。嗯，是的，nebula 是存储计算分离的，最好的好处是部署方式灵活，计算节点和存储节点可以根据不同的需求独立扩缩容。