3.3存储规范

知识的规则化输出结果主要是以图和表的形式，图知识和表知识有着不同的特性及结构，因此需要对它们分别设置不同的存储方案。以下对这两种形式知识的存储进行说明。

3.3.1 图知识的存储

图知识中有以下几个基本概念：图（Graph）指关系图。比如：同学及朋友关系图、银行转账图等；顶点（Vertex）一般指实体。比如：人、账户等；边（Edge）一般指关系。比如：朋友关系、转账动作等；属性（Property）顶点或边可以包含属性，比如：人的姓名、人的年龄、转账的时间。其逻辑形状如下图3-10所示。



图3-10 图知识逻辑形状

在图知识中最重要的信息就是图的节点、边的信息和边之间的关系，可以使用关系型数据库或图数据库来存储这些信息，具体如下。

图数据库是以图论为基础的一种非关系型数据库，其数据库存储结构和数据的查询方式都是以图论为基础的。图论中图的基本元素为节点和边，在图数据库中对应的就是节点和关系。图数据库的关注点是“关联关系”形成的图，其目标是对现实世界中的实体与实体之间的关联关系进行存储与分析：将实体抽象为顶点、将实体之间的关联关系抽象为边。通过顶点和边形成的图谱结构，直观自然地表达万物关联的世界，同时解决了复杂关联关系深层检索的性能问题。

图数据库仅使用一个模式和一组起点，就可以围绕这些初始起点探索相邻数据，收集和汇总来自数百万个节点和关系的信息，并保持搜索范围之外的任何数据不变。图数据库在处理规模大、关联度高的数据时优势明显。与传统的关系型数据库相比，图数据库具有如下优势：

（1）可以很自然地表达现实世界中的实体及其关联关系（对应图的顶点及边）；

（2）灵活的数据模型可以适应不断变化的业务需求；

（3）灵活的图查询语言，轻松实现复杂关系网络的分析；

（4）关系型数据库在遍历关系网络并抽取信息的能力非常弱，图数据库则为此而生；

（5）关系型数据库在规模庞大时很难做多层关联关系分析（Join操作往往消耗过长时间而失败），图数据库则天然把关联数据连接在一起，无需耗时耗内存的Join操作，可以保持常数级时间复杂度；

（6）在关系型数据库的查询中，关系的增多意味着需要遍历更多的关联表与实体表，其中存在着大量需要遍历的数据。并且，关系型数据库在数据查询时遍历的大量内容都与查询内容毫无关系，因此这也会导致关系查询时效率的大大降低；

（7）在进行数据查询时，图数据库会从查询目标的临近节点开始，基于图结构进行查询，而不是对整个类别的数据进行遍历。因此，在查询内容不变的情况下，数据规模的不断增大并不会对其查询性能产生明显影响；

（8）图数据库多用于网络安全、金融风控、知识图谱、广告推荐、社交网络、智能机器人、物联网等领域以及关联分析、路径搜索、子图挖掘、特征发现、社区聚类等应用。其主流的查询语言包括：Gremlin、Cypher、SPARQL，典型的存储结构有边集数组、邻接矩阵、邻接表和十字链表等。

综上所述，使用图数据库存储图数据能够产生更高的效率及性能。在使用图数据库存储图数据时，可以使用当前流行的图数据库Neo4j。Neo4j是一个由Java实现的开源图数据库，有效地将属性图模型实现到了存储级别，还提供了完整的数据库特性。它包括如下几个显著特点：完整的ACID支持、高可用性、轻易扩展到上亿级别的节点和关系、通过遍历工具高速检索数据。Neo4j确保了在一个事务里面的多个操作同时发生，保证数据的一致性。Neo4j把其数据库文件分为四大类来分类存储，分别是标签、节点、属性和关系。除此之外，Neo4j还提供了一个用户友好的Web页面，可以进行各项配置、写入、查询等操作，并且提供了可视化功能。Cypher是Neo4j的声明式图形查询语言，允许用户不必编写图形结构的遍历代码，就可以对图形数据进行高效的查询。Cypher的设计目的类似SQL，适合于开发者以及在数据库上做点对点模式（ad-hoc）查询的专业操作人员。其具备的能力包括：创建、更新、删除节点和关系；通过模式匹配来查询和修改节点和关系；管理索引和约束等。以上种种原因使得Neo4j最适合用于完整的企业部署或者用于一个轻量级项目中完整服务器的一个子集存在。

使用Neo4j图数据库存储图知识的步骤也非常简单方便：可以直接使用Cypher load csv语句将数据转换成CSV格式后通过LOAD CSV读取数据，也可以使用官方提供的neo4j-import工具进行批量导入存储。除了这两种方式之外，还可以使用官方提供的API或是一些别的导入工具进行存储。

3.3.2 表知识的存储

表知识中的各种数据被划分为一个个字段，每条数据对于同一个字段可能会有着不同的值，在存储表知识时可以使用关系型数据库。

关系型数据库（Relational Database）是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。现实世界中的各种实体以及实体之间的各种联系均用关系模型来表示。关系型数据库以行和列的形式存储数据，用户通过查询来检索数据库中的数据，而查询是一个用于限定数据库中某些区域的执行代码。关系模型可以简单理解为二维表格模型，而一个关系型数据库就是由二维表及其之间的关系组成的一个数据组织。

关系型数据库具有如下几个特点：

存储方式：传统的关系型数据库采用表格的储存方式，数据以行和列的方式进行存储，要读取和查询都十分方便。

存储结构：关系型数据库按照结构化的方法存储数据，每个数据表都必须对各个字段定义好（也就是先定义好表的结构），再根据表的结构存入数据，这样做的好处就是由于数据的形式和内容在存入数据之前就已经定义好了，所以整个数据表的可靠性和稳定性都比较高，但带来的问题就是一旦存入数据后，如果需要修改数据表的结构就会十分困难。

存储规范：关系型数据库为了避免重复、规范化数据以及充分利用好存储空间，把数据按照最小关系表的形式进行存储，这样数据的管理就可以变得很清晰、一目了然。

扩展方式：由于关系型数据库将数据存储在数据表中，数据操作的瓶颈出现在多张数据表的操作中，而且数据表越多这个问题越严重，如果要缓解这个问题，只能提高处理能力，也就是选择速度更快性能更高的计算机，这样的方法虽然可以一定的拓展空间，但这样的拓展空间一定有非常有限的，也就是关系型数据库只具备纵向扩展能力。

查询方式：关系型数据库采用结构化查询语言（即SQL）来对数据库进行查询，SQL早已获得了各个数据库厂商的支持，成为数据库行业的标准，它能够支持数据库的CRUD（增加，查询，更新，删除）操作，具有非常强大的功能，SQL可以采用类似索引的方法来加快查询操作。

规范化：在数据库的设计开发过程中开发人员通常会面对同时需要对一个或者多个数据实体（包括数组、列表和嵌套数据）进行操作，这样在关系型数据库中，一个数据实体一般首先要分割成多个部分，然后再对分割的部分进行规范化，规范化以后再分别存入到多张关系型数据表中，这是一个复杂的过程。随着软件技术的发展，相当多的软件开发平台都提供一些简单的解决方法，例如，可以利用ORM层（也就是对象关系映射）来将数据库中对象模型映射到基于SQL的关系型数据库中去以及进行不同类型系统的数据之间的转换。

事务性：关系型数据库强调ACID规则（原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）），可以满足对事务性要求较高或者需要进行复杂数据查询的数据操作，而且可以充分满足数据库操作的高性能和操作稳定性的要求。并且关系型数据库十分强调数据的强一致性，对于事务的操作有很好的支持。关系型数据库可以控制事务原子性细粒度，并且一旦操作有误或者有需要，可以马上回滚事务。

读写性能：关系型数据库十分强调数据的一致性，并为此降低读写性能付出了巨大的代价，虽然关系型数据库存储数据和处理数据的可靠性很不错，但一旦面对海量数据的处理的时候效率就会变得很差，特别是遇到高并发读写的时候性能就会下降的非常厉害。

在关系型数据库中，数据由不同的表组成，每个表包含有不同的字段，还可以对各个字段施加多种限制，其结构如下表3-9所示。

表3-9 关系型数据表结构

|  |  |  |
| --- | --- | --- |
| 字段1 | 字段2 | 字段3 |
| A | E | I |
| B | F | J |
| C | G | K |

在使用关系型数据库存储图数据时，可以使用当前流行的MySQL数据库管理系统。在Web应用方面，MySQL是最好的RDBMS （Relational Database Management System，关系数据库管理系统） 应用软件之一。MySQL所使用的SQL语言是用于访问数据库的最常用标准化语言。MySQL软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择MySQL作为网站数据库。

使用MySQL数据库存储表知识也是非常方便快速：提前对表知识进行一些规范化处理后将其存储为CSV文件、EXCEL文件等格式，然后可以使用各种各样的数据库管理工具从这些文件中批量导入数据即可。

3.3.3 存储工具推荐

综上所述，图知识和表知识有着不同的特性和结构，其存储方式、存储工具等都不相同，对它们的存储既要符合它们独特的数据结构，又要能够在查询时方便地进行检索。关于图知识和表知识的特点、可使用的存储工具及建议使用的较成熟的数据库工具如下表3-10所示：

表3-10 图知识和表知识特点及建议存储工具

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 特点 | 可使用的存储工具 | 建议使用 |
| 图知识 | 数据以节点和边的形式展现出来 | 图数据库、关系型数据库 | Neo4j图数据库 |
| 表知识 | 数据以二维表格的形式展现出来 | 关系型数据库 | MySQL数据库 |

3.3.4 统一格式转换

由于风洞侧知识库中知识格式的差异，导致各个知识库中知识的数据结构格式不同、信息量大且易存在质量问题，增大了知识工程师处理与分析的难度。若对风洞侧的知识格式进行统一后再存入中心侧知识库，就能够对海量知识数据实现“存、管、用”的一体化处理方案，可以有效增强系统的数据标准化支撑能力、数据及时性、完整性、准确性等。数据格式统一的示意图如下图X所示：



图X 知识格式统一示意图

由于知识规则化输出主要为图知识和表知识，因此这里主要考虑对图知识和表知识格式的统一处理。表知识的存储可以使用关系型数据库，其技术及管理工具已经较为成熟，因此在存储之前先把图知识转化为表知识的形式，然后统一以表知识的形式进行存储是一个很好的方式。

在使用表结构存储图知识的时候，应该为其建立两个不同的数据表。第一个表用来存储图知识中的节点及其属性信息，在其中需要设置node\_id字段表示节点的唯一标识，如果有相关属性则额外建立对应字段表示其属性值；第二个表用来存储节点之间的关系信息，设置relation\_id字段表示关系的唯一标识，设置node1\_id字段和node2\_id字段表示与该关系相关的两个节点的node\_id，设置relation字段具体表示其关系。

上图3-10所示的图知识可以建立如下表X和表X所示的两个数据表存储其信息。

表X 存储图知识的节点信息的数据表

|  |  |
| --- | --- |
| node\_id | node |
| 1 | 实体A |
| 2 | 实体B |
| 3 | 实体C |

表X存储图知识的关系信息的数据表

|  |  |  |  |
| --- | --- | --- | --- |
| relation\_id | node1\_id | node2\_id | relation |
| 1 | 1 | 2 | 关系AB |
| 2 | 1 | 3 | 关系AC |
| 3 | 2 | 3 | 关系BC |

在图知识转化为表知识的过程中，应注意以下几点：

（1）对原有图知识中节点、属性及关系的名称应该保留，不应随意设置新的名称。

（2）在表示关系信息的数据表中应该使用节点的唯一标识node\_id表示节点，不应直接使用节点的名称。

（3）在表示节点或关系的数据表中，如有必要可以增加备注字段，以表示该节点或关系的具体信息。

在建立了相关数据表并存储相关信息之后，对原有图知识的存取即转化为对表知识的存取，实现了知识的统一格式，可以使用关系型数据库对知识整体进行统一存取。