软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006180]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

PandaDB:一种异构数据智能融合管理系统*

沈志宏[1], 赵子豪[1,2], 王华进[1], 刘忠新[1], 胡 川[1,2], 周园春[1,2]

[1](中国科学院计算机网络信息中心 北京 100190)

[2](中国科学院大学 北京 100049)

通讯作者: 沈志宏, E-mail: bluejoe@cnic.cn

摘 要: 随着大数据应用的不断深入,对大规模结构化/非结构化数据进行融合管理和分析的需求日益凸显.然而,结构化/非结构化数据在存储管理方式、信息获取方式、检索方式方面的差异给融合管理和分析带来了技术挑战.本文提出了适用于异构数据融合管理和语义计算的属性图扩展模型,并定义了相关属性操作符和查询语法.接着,基于智能属性图模型提出异构数据智能融合管理系统 PandaDB,并详细介绍了 PandaDB 的总体架构、存储机制、查询机制、属性协存和 AI 算法集成机制.性能测试和应用案例证明,PandaDB 的协存机制、分布式架构和语义索引机制对大规模异构数据的即席查询和分析具有较好的性能表现,该系统可实际应用于学术图谱实体消歧与可视化等融合数据管理场景.

关键词: 数据管理系统;异构数据融合;图数据模型;即席查询;人工智能.

中图法分类号: TP311

中文引用格式: 沈志宏,赵子豪,王华进,刘忠新,胡川,周园春.PandaDB:一种异构数据智能融合管理系统.软件学报. http://www.jos.org.cn/1000-9825/6180.htm

英文引用格式: Shen ZH, ZHAO ZH, Wang HJ, Liu ZX, Hu C, Zhou YC. PandaDB: An intelligent management system for heterogeneous data. Ruan Jian Xue Bao/Journal of Software, 2021 (in Chinese). http://www.jos.org.cn/1000-9825/6180.htm

PandaDB: An Intelligent Management System for Heterogeneous Data

SHEN Zhi-Hong[1], ZHAO Zi-Hao[1,2], WANG Hua-Jin[1], LIU Zhong-Xin[1], HU Chuan[1,2], ZHOU Yuan-Chun[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: With the development of big data application, the demand of large-scale structured/unstructured data fusion management and analysis is becoming increasingly prominent. However, the differences in management, process, retrieval of structured/unstructured data brings challenges for fusion management and analysis. This paper proposes an extended property graph model for heterogeneous data fusion management and semantic computing, defines related property operators and query syntax. Based on the intelligent property graph model, this paper implements PandaDB, an intelligent heterogeneous data fusion management system. This paper depicts the architecture, storage mechanism, query mechanism, property co-storage, AI algorithm scheduling and distributed architecture of PandaDB. Test experiments and cases show that the co-storage mechanism and distributed architecture of PandaDB have good performance acceleration effects, and can be applied in some scenarios of fusion data intelligent management such as academic knowledge graph entity disambiguation.

Key words: Data Management System; Heterogeneous data fusion; Graph Data Model; Ad-hoc Query; AI.

* 基金项目: 中国科学院战略性先导科技专项 B 类课题(XDB38030300);国家自然科学基金重点项目((61836013);科技部创新方法工作专项(2019IM020100),中国科学院信息化专项课题(XXH13503)

Foundation item: Strategic Priority Research Program of CAS (XDB38030300); Key Project of National Natural Science Foundation of China(61836013); Ministry of Science and Technology Innovation Methods Special work Project under grant (2019IM020100); Informatization Plan of Chinese Academy of Sciences(XXH13503)

收稿时间: 2020-07-20; 修改时间: 2020-09-03; 修改时间: 2020-11-06; jos 在线出版时间: 2021-01-20

1 引言

在大数据时代,随着各类应用的推广使用,数据产生速度越来越快、数据体量越来越大.一方面,数据采集技术的迅猛发展使得数据的结构更多样、种类更丰富.数据表现出多元异构的特点,非结构化数据在其中占有较大比重.有研究表明,视频、音频、图片等非结构化数据占据高达 90%的比例[1]；另一方面,近年来,数据中台、知识图谱等数据管理分析技术得到了广泛的应用.数据中台要求结构化/非结构化数据能够在统一的环境中得到良好的治理,以便支持多种应用；知识图谱,特别是多模态知识图谱[2],要求对底层结构化/非结构化数据进行融合关联分析,并支持用户进行交互式查询.这些技术均提出对结构化/非结构化数据进行融合管理和分析的需求.

结构化数据通常具有较为规范、统一的形式.目前,针对结构化数据的管理和分析,已具有成熟的数据模型、查询语言和管理系统.与结构化数据相比,非结构化数据的管理方式存在着诸多差异,这给高效的结构化/非结构化数据的融合管理和分析带来了多方面的挑战:

(1) **分离的存储管理方式,给结构化/非结构化数据的统一管理带来挑战.**相对于结构化数据,非结构化数据占有更大的空间,出于读写效率考虑,非结构化数据往往单独存在于文件系统,或者对象存储系统,这使得维护结构化/非结构化数据一致性的难度增大；

(2) **差异化的信息获取方式,给结构化/非结构化数据的统一分析带来挑战.**相对于结构化数据,非结构化数据内容比较复杂,为了实现高效检索和分析,往往需要预先引入模式识别、深度学习等方法实现信息抽取和数据挖掘,从而获取非结构化数据所蕴含的内在信息；

(3) **不一致的检索方式,给结构化/非结构化数据的一致化即席查询带来挑战.**与结构化数据具有较为成熟的 SQL、类 SQL 查询语言的现状不同,非结构化数据的信息检索往往缺乏统一的操作模式和查询语法,目前采用的多是逐案的个性化方案.

为实现结构化/非结构化数据的融合管理和分析,需要从模型层面出发,设计统一的表示和查询方法.传统的关系模型、属性图模型不能有效揭示和表示非结构化数据的内在信息.有学者提出将数据和 Schema 表示为边标记图,以此替代非结构化数据底层类型约束的缺失[3].但该方法仅提出一种为非结构化数据添加 Schema 的方法,不能实现对非结构化数据中信息的自由检索.Li 等人提出从基本属性、语义特征、底层特征和原始数据等四个角度定义非结构化数据[4],但这种方法依赖于预定义,并不适用于非结构化数据的交互查询.近年来有学者提出在非结构化数据流上抽取 RDF 三元组的方法[5],该方法只实现了三元组的抽取,不能支持对非结构化数据内在信息的交互式查询,且并不具备数据管理系统的基本能力.

另外一种融合管理的路线是将非结构化数据在数据库中存储为二进制大对象（BLOB,Binary Large OBject）,当应用获取数据的时候,返回一个二进制数组或者数据流.这种方法在性能和功能上都不令人满意[6].针对此问题,研究人员提出了一系列非结构化数据管理系统[7][8][9],这些系统综合考虑了非结构化数据体积大、结构复杂的特点,设计了合适的存储模型,一定程度上解决了非结构化数据的存储和管理问题,但其提供的查询服务仅基于文件对象本身和元数据,不能提供对非结构化数据内在信息的查询能力.

由此,本文提出属性图扩展模型及其查询方法.属性图扩展模型在传统属性图的基础增加了对非结构化数据内在信息的表示能力,以及结构化和非结构化数据之间的互操作能力.在此基础上,本文继而提出基于智能属性图模型的异构数据智能融合管理系统 PandaDB.

本文在第 2 节给出属性图扩展模型和相关概念,包括层叠属性图、智能属性图、次级属性等,并提出属性操作符和查询语法.在第 3 节中给出 PandaDB 的系统设计与具体实现.在第 4 节中通过实验和案例验证该系统的效率及可行性.第 5 节介绍与本文研究相关的工作.最后,对未来研究可能面临的挑战进行展望.

2 概念设计

传统属性图模型无法有效表示非结构化属性,本节提出属性图扩展模型,以解决非结构化属性的有效表示

问题,然后介绍针对属性图扩展模型的语义操作和查询语法设计,以支持因引入非结构化属性及其内在信息所带来的新的查询特性.

2.1 属性图扩展模型

传统属性图模型可以形式化表示为 $G=(V,E,P)$,其中 G 表示全体数据,V 表示数据中的实体集合,E 表示实体间的关系集合,P 表示数据集中实体的属性集合.

针对图片、语音、文本这样的非结构化属性,属性图模型无法有效揭示其蕴含的内在信息（如:某类顶点的 photo 属性蕴含有“车牌号”信息）,内在信息通常是离线且具有结构延迟的:

- **离线**:将非结构化属性转化为结构化、半结构化属性的信息抽取过程,属于对数据的预处理;
- **结构延迟:**非结构化属性的内在信息不具备明确定义的结构,而是根据应用的后期需要从而选择特定的信息抽取方法,这种结构是不明确的、延迟定义的.

为增强对结构化属性、非结构化属性的统一表示能力,本文针对属性图模型进行了扩展,提出层叠属性图模型和智能属性图模型.

定义 2-1: 具备以下特征的属性图被称为层叠属性图(Cascading Property Graph):

1) 层叠属性图 G^c 可以表示为 $G^C=(V,E,PP,PN)$,其中 PP 是基本属性（Primitive Property）集合,PN 是内嵌式属性（Nested Property）集合;
2) 基本属性的值为文本、数值、二进制数组等基本数据类型;
3) 内嵌式属性的值为另外一个属性图;

定义 2-2: 具备以下特征的属性图被称为智能属性图（Intelligent Property Graph）:

1) 智能属性图 G^I 可以表示为 $G^I=(V,E,PP,O)$,其中 O 为语义操作集合;
2) 存在 $PI \subseteq PP$, PI 为智能属性集合;
3) $O=(S_\varphi,S_\xi)$,其中 S_φ 为智能属性展开操作集合, S_ξ 为智能属性语义计算操作集合;
4) 对于智能属性 $p_i \in PI$,存在展开操作 $\varphi \in S_\varphi$,可满足 $p_n=\varphi(p_i)$,且 $p_n \in PN$,即将智能属性 p_i 展开为内嵌式属性 p_n;
5) 对于智能属性 $p_1 \in PI$ 和 $p_2 \in PI$,存在语义计算操作 $\xi \in S_\xi$,可满足 $\sigma=\xi(p_1,p_2)$ 对 p_1 和 p_2 两个智能属性进行语义计算,得到结果 σ.

定义 2-3: 内嵌式属性具有次级属性（Sub-property）:

1) 对于智能属性图 $G^I=(V,E,PP,O)$,若存在 $V_i \in V$ 且 V_i 具有内嵌式属性 p_i^N ,则根据定义 2-1 有 $p_i^N=G_i^N$,其中 G_i^N 为传统属性图 (V_i^N,E_i^N,P_i^P);
2) 若 $|V_i^N|=1$,则 P_i^P 的任一元素 P_{ij}^S 称为 V_i 的次级属性.

作为示例,图 1 示出一个智能属性图,图中存在一个 Car 类型的顶点 car1,其具有一个智能属性 photo,执行展开操作后,photo 具有两个次级属性:photo->plateNumber 和 photo->model.

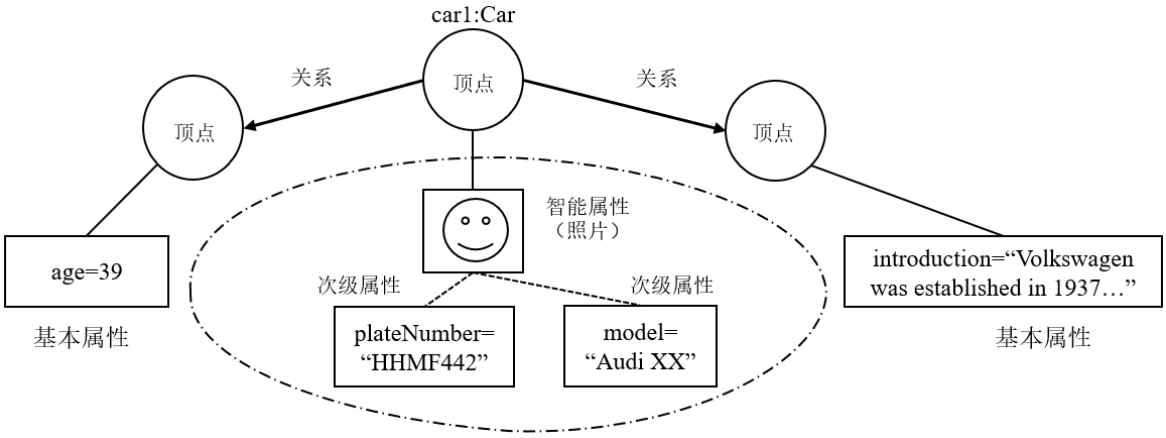

Fig.1 Intelligent Property Graph Model

图 1 智能属性图模型

2.2 属性操作符

基于定义 2-2,本文定义了针对智能属性的次级属性抽取操作符和语义计算操作符:

- **次级属性抽取操作符:**次级属性抽取操作符“->”用以抽取智能属性的次级属性,如针对 photo 属性执行“photo->plateNumber”,即可获取到 photo 中的车牌号;
- **语义计算操作符:**传统属性图查询语言中的谓词（Predicate）,只支持对属性进行比较,如=、>、<、正则匹配等.本文针对智能属性新增~:、::、:>等拓展谓词,以表示非结构化属性之间的相似关系、相似度、包含关系等逻辑.

Table.1 Semantic Computation Operators

表 1 语义计算操作符

操作名称	符号	含义	示例
SemanticCompare	::	计算 x 和 y 之间的相似度	x::y=0.7
SemanticLike	~:	计算 x 和 y 是否相似?	x~:y=true
SemanticUnlike	!:	计算 x 和 y 是否不相似?	x!:y=false
SemanticIn	<:	计算 x 是否在 y 里	x<:y=true
SemanticContain	>:	计算 x 是否包含 y	y>:x=true

在属性图模型中,由于内在信息的不可见性,非结构化属性之间的计算操作支持有限.结合定义 2-1、2-2、2-3 中关于非结构化属性的表述和属性操作符的特性,智能属性图模型可以实现非结构化属性内在信息的在线获取,且结构是预定义的.

- **在线**:从非结构化属性中获取结构化、半结构化信息（即次级属性）的过程是按需的,无需对非结构化属性进行专门的预处理;
- **结构预定义**:非结构化数据内在信息依赖于 schema 层面的定义,而非依赖于信息抽取工具的实现.在底层查询机制的支持下,可以实现针对非结构化属性的直接运算.

2.3 查询语法

本文针对智能属性图模型,针对标准化 Cypher 查询语言[10]进行扩展,形成 CypherPlus 语言.CypherPlus 定义了 BLOB 属性类型,用以表示非结构化属性的值.CypherPlus 同时引入了 BLOB 字面值、次级属性抽取操作符、语义计算操作符等新特性,从而支持对非结构化属性的表示和语义操作.

(1) BlobLiteral:用于表示非结构化属性的字面值,格式如<schema://path>,其中 schema 可以为 FILE、HTTP(S)、FTP(S)、BASE64 等多种类型.如图 2 所示;

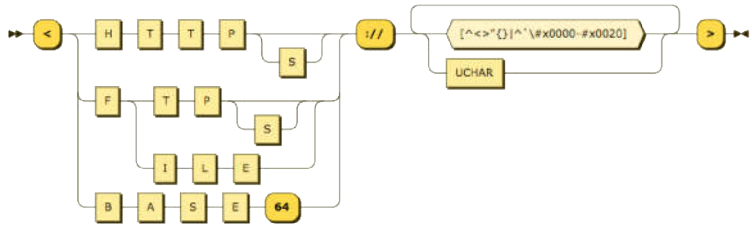

Fig.2 Grammer definition of BlobLiteral

图 2 BlobLiteral 语法定义

(2) SubPropertyExtractor:用于表示次级属性的抽取操作,如图 3 所示,其中 PropertyKeyName 为次级属性的名称;

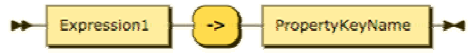

Fig.3 Grammer definition of SubPropertyExtractor

图 3 SubPropertyExtractor 语法定义

(3) SemanticComparison:语义计算操作符,包括 SemanticCompare、SemanticLike、SemanticUnlike、SemanticIn、SemanticContain 等操作符.以 SemanticLike 为例,它用以指示两个属性的值是否相似,语法定义如图 4 所示,其中 AlgorithmName 为指定计算的算法名称,Threshold 为阈值,AlgorithmName 和 Threshold 为可选项,该情况下执行引擎则采用默认的比较器和阈值.

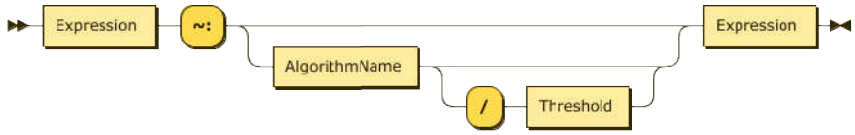

Fig.4 Grammer definition of SemanticLike

图 4 SemanticLike 语法定义

例如,针对图 1 的数据模型,可以查找与车牌号为 HHMF442 的车相似的车,查询语句如下:

Q1: match (c1:CAR), (c2:CAR) where c1.photo~:c2.photo and c1->plateNumber='HHMF442' return c2

Q2: return 'Zhihong SHEN' ::jaro 'SHEN Zhihong'

查询语句 Q2 用以计算两个文本的相似度.

3 系统实现

为实现结构化、非结构化数据的融合管理和关联查询分析,本文采用智能属性图模型,基于 Neo4j 开源版本,设计并实现了异构数据智能融合管理系统 PandaDB.本部分 3.1 小节介绍 PandaDB 的总体架构,3.2 小节到 3.5 小节分别介绍各模块的设计思路和实现细节.

3.1 总体架构

PandaDB 基于智能属性图模型组织数据,底层数据被分为图结构数据、结构化属性数据和非结构化属性数据三部分.其中,图结构数据指图的节点和边等描述图结构的数据；结构化属性数据指数值、字符串、日期等类型的数据；非结构化属性数据泛指除结构化数据之外的数据,如视频、音频、图片、文档等.PandaDB 以 BLOB 对象的形式存储非结构化数据,并将其表示为实体（节点）的属性.根据上述三类数据的应用特点,PandaDB 设计了分布式多元存储方案:

- 分布式图数据存储:基于传统的图数据库保存图结构数据和属性数据,在每个节点上保存相同的数据副本；
- 结构化属性协存:基于 ElasticSearch、Solr 等外部存储实现大规模结构化属性数据的存储和索引构建；
- BLOB 存储:基于 HBase、Ceph 等存储系统实现非结构化属性数据的分布式存储；

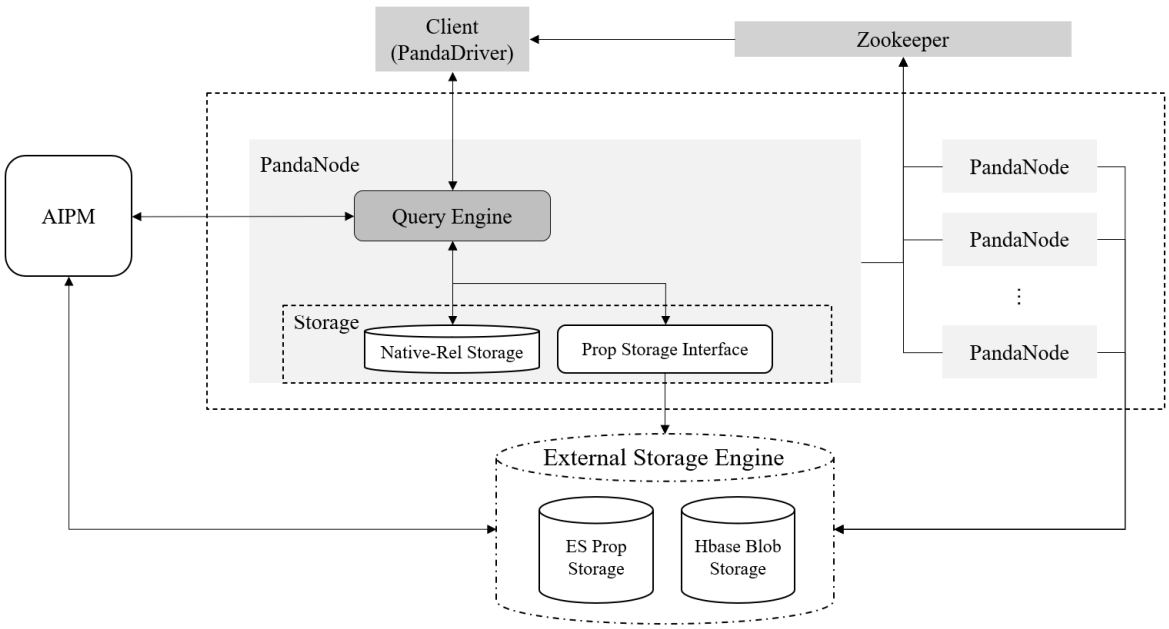

Fig.5 Architecture of PandaDB

图 5 PandaDB 总体架构设计

PandaDB 总体架构如图 5 所示,重要模块描述如下:

- 存储引擎:维护本地的图结构数据,调度外部属性存储,按需为查询引擎提供服务；
- 外部存储:包括基于 ElasticSearch 的结构化属性协存和基于 HBase 的 BLOB 存储两部分；
- 查询引擎:解析并执行 CypherPlus 查询；
- AIPM:AI 模型服务框架,通过模型和资源管理,实现 AI 模型的灵活部署、高效按需运行,同时有效屏蔽 AI 模型之间的依赖.

PandaDB 集群采用了无主架构设计,图 5 中 PandaNode 是其中的节点,包含查询引擎和存储引擎两部分.属性数据和非结构化数据存储在外置分布式存储工具中,PandaNode 只保存图结构数据,通过属性存储接口与外置存储交互.

3.2 存储机制

PandaDB 将 BLOB 引入了 Neo4j 的类型系统,同时对 Neo4j 的存储结构进行了改造.存储结构如图 6 所示,

除了 Neo4j 保留区,BLOB 的属性字段同时记录了 BLOB 的元数据,包括:唯一标识 blobid、长度 length 和 MIME 类型.

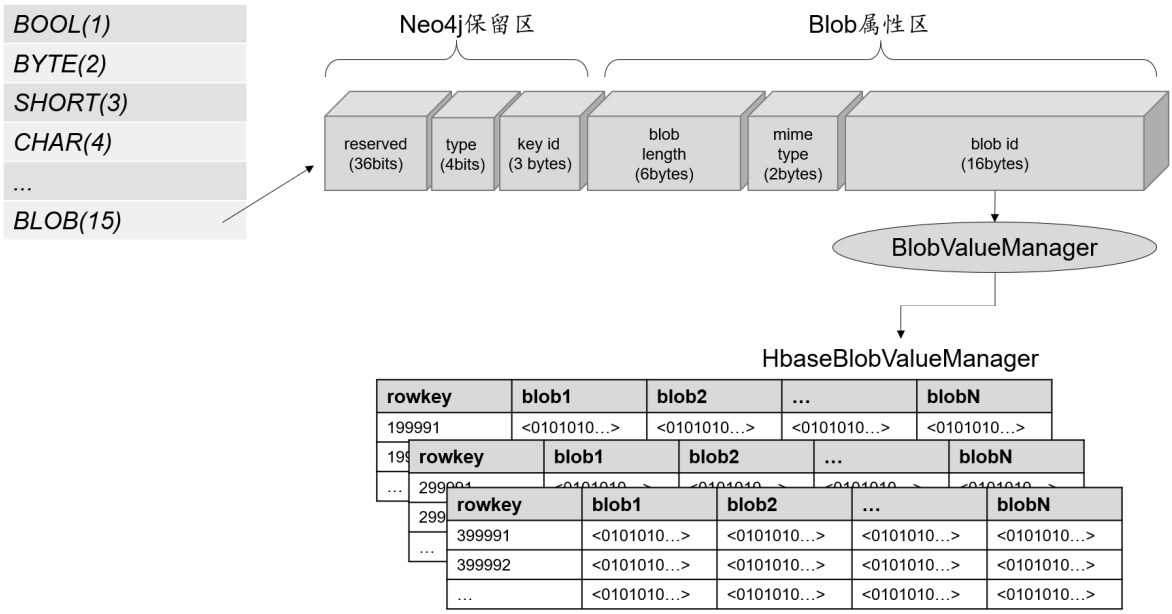

Fig.6 Design of BLOB storage structure

图 6 BLOB 存储结构设计

为实现对外部 BLOB 存储系统的调用,PandaDB 中设计了 BlobValueManager 接口,定义了 getById()/store()/discard()等操作方法.作为 BlobValueManager 的一个实现,HBaseBlobValueManager 基于 HBase 集群实现 BLOB 数据的存取.在该方案中,为支持大规模 BLOB 的存储,HBase 被设计成包含 N 列的宽表,采用 blobid/N 作为 HBase 表的 rowkey,blobid%N 对应于 HBase 的某一列.

为加速 BLOB 的读取,BLOB 的内容读取被封装为一个 InputStream,在用户通过 Bolt 协议获取 BLOB 内容或进行语义计算的时候,这种流式读取的机制提高了运行的性能.

多元存储带来了存储事务保证的复杂性,客户端在写入数据时,将写操作请求发送到 PandaDB 的 Leader 节点,然后由 Leader 节点执行具体的写入操作.Leader 节点的具体操作流程如下:

(1) Leader 节点开启事务,执行 Cypher 解析,翻译成具体的执行操作;

(2) 向 BLOB 存储引擎发送请求,执行 BLOB 数据的写入操作.若执行失败,则向上回滚,标记事务失败;

(3) 若 BLOB 数据写入成功,则执行图结构数据和结构化属性数据的写入操作.若执行失败,则向上回滚,标记事务失败;

(4) 将结构化属性数据的修改,同步到协存.若执行失败,则向上回滚,标记事务失败;

(5) 执行事务提交;

(6) 关闭事务,返回操作成功.

3.3 查询机制

PandaDB 查询引擎主要实现查询语句的解析、逻辑计划的生成与优化、物理计划的优化与执行.基于 Neo4j,PandaDB 查询引擎主要改进如下几个部分:

(1) 解析阶段:增强 Cypher 语言的解析规则,支持 BLOB 字面常量(BlobLiteral)、BLOB 次级属性的抽

取操作符（SubPropertyExtractor）,以及属性语义操作符（SemanticComparison）;

(2) 语法检查阶段:针对 BlobLiteral、SubPropertyExtractor、SemanticComparison 执行形式检查,如发现非法的 BLOB 路径,非法的语义算子和阈值等;

(3) 计划优化阶段:针对 BlobLiteral 的操作进行优化,针对大规模属性过滤情形,采用谓词下推策略等;

(4) 计划执行阶段:充分调度属性协存模块、AIPM 模块,以及 BLOB 存储模块,实现高效的属性优先过滤、BLOB 获取与语义计算; 图 7 示出了一个典型的查询过程,该查询要求返回所有与 photo0 中人脸相似,且 age 值大于 30 的节点.

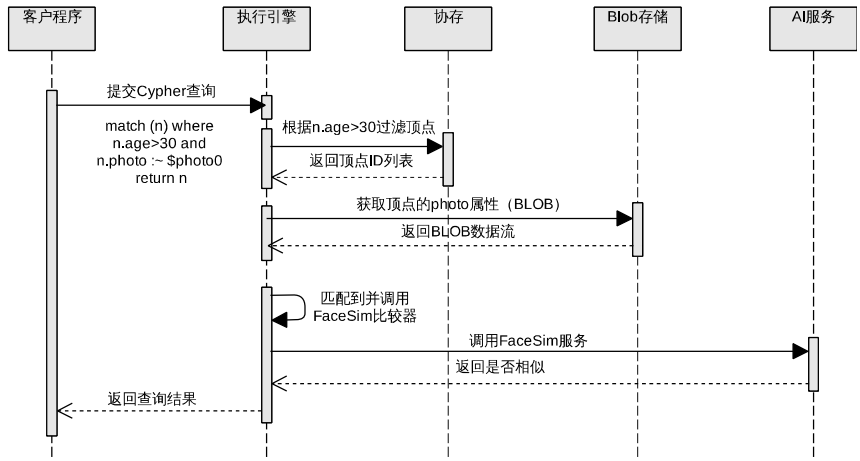

Fig.7 CypherPlus Query Process

图 7 CypherPlus 查询流程

图 8 从查询语法、查询计划、执行引擎三个层面示出 PandaDB 查询机制的设计.解析引擎将查询语句中的符号转化为语义算子,执行引擎根据规则集选择 AI 模型处理对应的数据,并将结果返回.

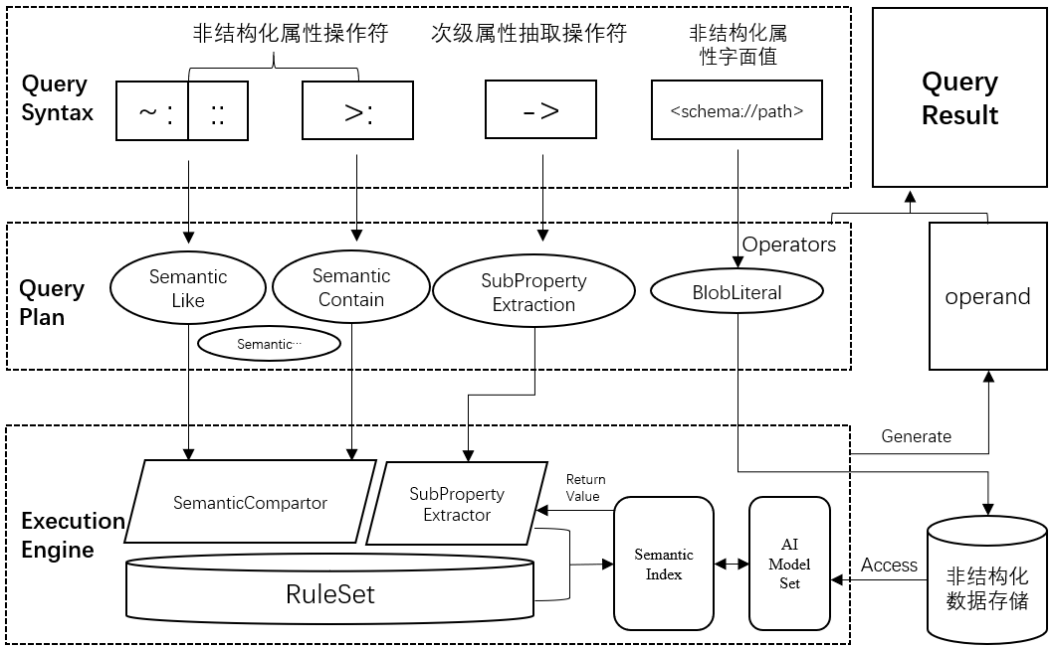

Fig.8 Query Mechanism of intelligent property graph

图 8 智能属性图查询机制

为加速非结构化数据的查询,PandaDB 实现了语义索引功能.非结构化数据中的信息被认为是一种语义信息,如图片中的人脸、图片中汽车的车牌号、录音中所包含的文字信息等.AI 模型从非结构化数据中抽取信息的过程可以看作是数据从高维空间到低维空间的映射,在低维空间映射的结果可以作为数据在该场景下的语义索引.

如在人脸比对查询的过程中,需要比较不同图片中人脸的相似度,通常的做法是利用人脸识别模型抽取人脸特征,比较两个特征的相似度.在 PandaDB 中,以向量形式表示的人脸特征被视为该非结构化在当前查询场景下的语义索引.系统处理涉及人脸比对的查询时,优先检查是否有对应的语义索引,若该查询对应的语义索引存在,则不向 AIPM 发起处理请求,直接比较语义索引得到结果.

语义索引可以减少查询引擎对 AI 服务的请求次数,从而避免了重复的数据传输,有利于提高系统效率.

3.4 属性数据协存

PandaDB 引入属性数据协存机制,用于实现结构化属性数据的全文索引,提高节点属性的过滤查询效率.目前,PandaDB 支持 ElasticSearch 作为协存引擎.

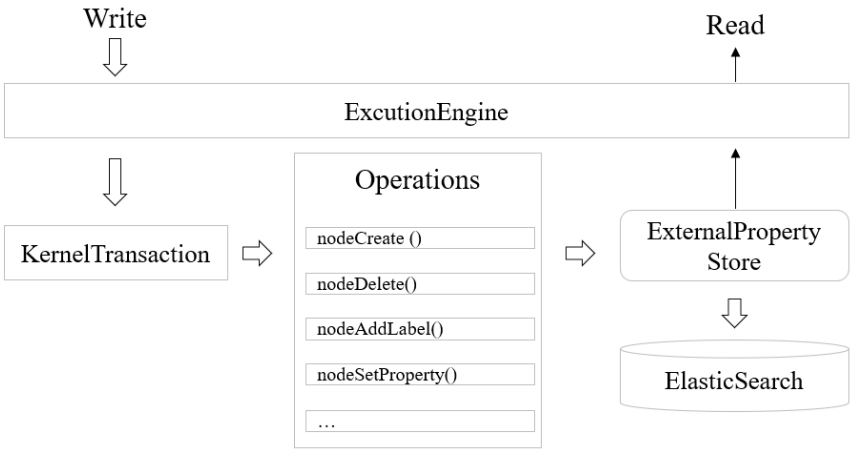

Fig.9 Write progress of property co-store module in PandaDB

图 9 PandaDB 协存模块的写入流程

图 9 示出 PandaDB 协存模块的写入流程,协存机制的关键设计如下:

(1) 属性数据在 ElasticSearch 中的存储结构:每个 Neo4j 图数据库对应协存引擎(ElasticSearch)中一个独立的索引,每个节点的属性数据和标签数据均组织成 ElasticSearch 中的一个文档,其中节点在 Neo4j 数据库中的 ID 作为文档的 ID,节点属性名作为文档的属性标签,节点属性值作为文档的属性数据,节点的标签数据表示为特殊设置的属性标签.数值、字符串、坐标、日期、时间等结构化属性数据类型分别转换为 ElasticSearch 中的对应数据类型;

(2) 属性写入及更新:为了维持 Neo4j 数据库中的本地数据和 ElasticSearch 中数据的一致性,PandaDB 对 Neo4j 中事务操作执行模块(Operations)中的节点更新部分进行了扩展,设计了 ExternalPropertyStore 用于存储在 Neo4j 事务中执行的所有操作.当 Neo4j 数据库执行插入节点、添加标签、设置属性、删除节点等操作的同时,也将对应的操作数据缓存到 ExternalPropertyStore 中,当 Neo4j 数据库执行事务提交操作的同时将缓存的操作数据同步到 ElasticSearch 中;

(3) 属性过滤:为了基于协存实现节点属性过滤,PandaDB 对 Neo4j 的 Cypher 查询执行计划进行了改造,

将节点属性过滤谓词下推到协存管理模块.根据谓词过滤条件生成 ElasticSearch 的检索请求,最后将命中的文档（节点）列表返回给查询引擎做进一步筛选.为了避免大量查询结果增大网络传输延迟,PandaDB 采用了异步分批的方式传递的查询结果.

3.5 AI算法集成与调度

AI 算法集成与调度主要包括本地算法驱动管理和 AI 算法服务框架.

本地算法驱动管理:为了对不同的抽取器（SubPropertyExtractor）和语义比较器（SemanticCompartor）进行统一管理,PandaDB 制定了驱动管理规则库.图 10 是对规则库中部分内容的展示,其中 DogOrCatClassifier 用以抽取宠物类型,适用于 blob/image 类型的属性；CosineStringSimilarity 用以计算两个文本串的余弦相似度,仅适用于两个 string 类型的属性.

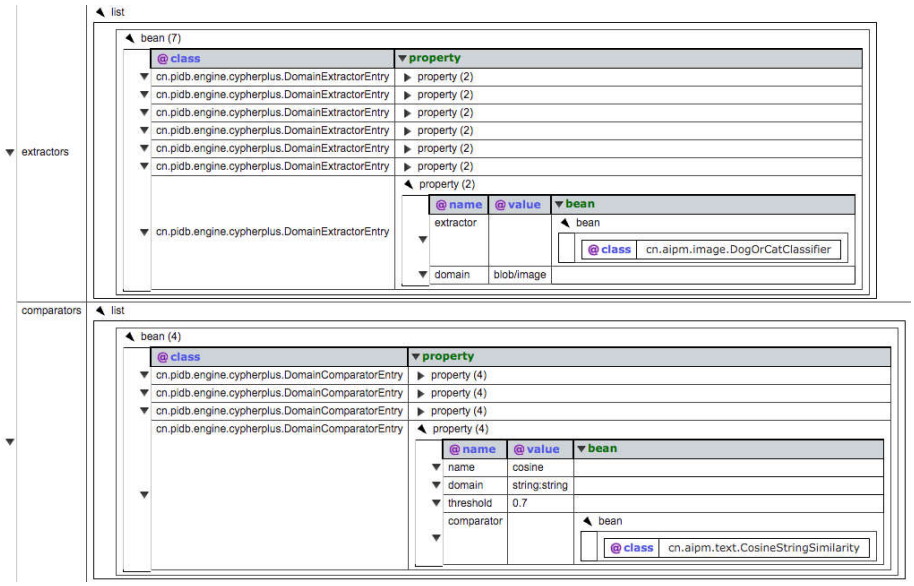

Fig.10 Extractor and Semantic comparator matching rule set

图 10 抽取器和语义比较器匹配规则库

AI 算法服务框架: PandaDB 中的信息抽取能力通过 AI 服务方式实现,AIPM 为 PandaDB 提供 AI 服务,它可以屏蔽不同 AI 模型之间的依赖冲突问题,降低人工智能模型的部署和维护难度,便于 PandaDB 按需扩展 AI 算子.图 11 给出了 AIPM 与系统之间的交互逻辑.系统以 HTTP 请求的方式向 AIPM 发出查询请求,请求的路径与 AI 算法具有对应关系,AIPM 接受到查询请求,调用对应的 AI 算法处理数据,以 JSON 字符串的形式返回结果.为了增强 AI 算子的可扩展性,AIPM 设计了统一的集成接口,并要求算子提供对这些接口的支持.图 12 是 AI 模型管理框架的示意图.

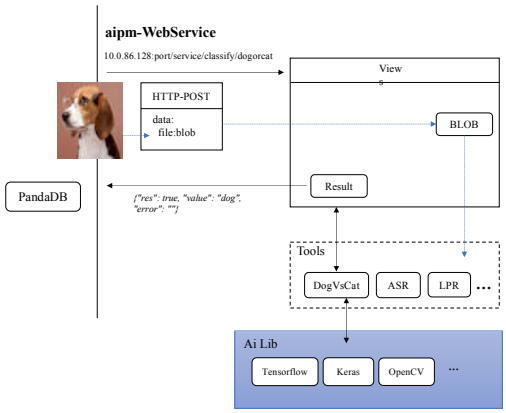

Fig.11 Interaction of PandaDB and AIPM

图 11 AIPM 与 PandaDB 交互框架

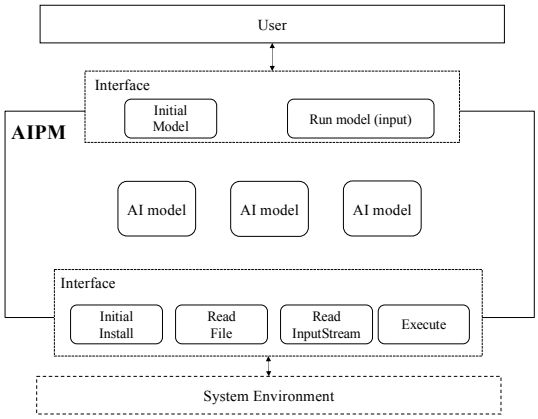

Fig.12 AI model management Framework

图 12 AI 模型管理框架

4 系统效果评估

为验证模型设计及 PandaDB 实现的有效性,本文针对属性协存、分布式方案、非结构化数据信息查询进行了测试,同时通过应用案例验证 PandaDB 对结构化和非结构化数据的融合管理能力.

4.1 属性协存性能测试

本测试验证基于 ElasticSearch 属性协存方案的性能,对 Neo4j 和引入协存方案后的 PandaDB 的查询性能进行对比,测试环境如表 2 所示.

Table.2 Information about test environment

表 2 测试环境信息

测试环境	环境描述
服务器软硬件环境	5 台同等配置的物理服务器,单台服务器配置为: CPU:32 颗 Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz 内存:128GB 硬盘:6TB 7200 RPM SAS HDD 网络:1000 mbps 互联 操作系统:CentOS Linux release 7.7.1908 (Core)
测试软件版本	Neo4j:3.5.6 （社区版） PandaDB:v0.0.2（开发版） ElasticSearch:6.5.0
环境部署	5 台服务器均部署了 ElasticSearch； 4 台服务器部署 PandaDB； 1 台服务器部署 Neo4j；（注:该 Neo4j 版本仅支持单节点部署）
测试数据	图数据集:1 亿节点（包含 6 种标签）,17 亿关系（包含 10 种类型） （注:该数据集为基于科研领域人才、机构和论文、奖励、专利、标准、专著等成果建立的真实数据集）

实验中使用的 Cypher 查询语句如表 3 所示.实验中对每条查询语句分别进行多次测试并取执行时间的平均值.为避免冷启动带来的性能影响,测试前对系统进行了预热.

Table.3 Query statement in propery co-store test

表 3 协存方案验证测试的查询语句

编号	测试语句	测试类型描述
Q-1	match (n:paper) where n.country="Malta" return count(n) ;	节点查询（单属性精准过滤）
Q-2	match (n:person) where n.org STARTS WITH "Shanghai" return count(n);	节点查询（单属性模糊匹配）
Q-3	match (n:paper) where n.country = "United States" and n.citation = 10 return count(n) ;	节点查询（双属性精准过滤）
Q-4	match (n:person) where n.org STARTS WITH "Shanghai" and n.citations=1 return count(n);	节点查询(模糊匹配与精准匹配结合的双属性过滤）
Q-5	match (n:person) where n.citations=100 and n.citations5=200 return count(n);	节点查询（双属性精准过滤）
Q-6	match (n:person) where n.citations=192 and n.citations5=204 and n.nationality="France" return count(n);	节点查询（三属性精准过滤）
Q-7	match (n:person) where n.citations=192 and n.citations5=204 and n.nationality="France" and n.publications5=5 return count(n);	节点查询（四属性精准过滤）
Q-8	match (startNode:paper)-[r]->(other) where startNode.country = "Japan" and startNode.citation = 5 return count(other);	关系查询（返回末端节点信息）
Q-9	match (n:person)-[r:work_for]->(other) where n.org starts with "Beijing" and n.citations = 1 return count(r)	关系查询（返回关系信息）
Q-10	match (startNode: person { personId:'32'}) optional match (startNode)-[r:write_person] -(other) return count(other)	关系查询（可选关系匹配）

Table.4 Test result for property co-store

表 4 协存方案验证测试结果

测试语句编号	平均执行时间（单位:ms）		执行时间比值
	Neo4j	PandaDB	
Q-1	184	189	0.97
Q-2	96.5	97.5	0.98
Q-3	1001.5	55	18.20
Q-4	426	42	10.14
Q-5	358	309	1.15
Q-6	304.5	51.5	5.91
Q-7	94	52.5	1.79
Q-8	2,658	2,210	1.20
Q-9	453	415	1.09
Q-10	53	62	0.85

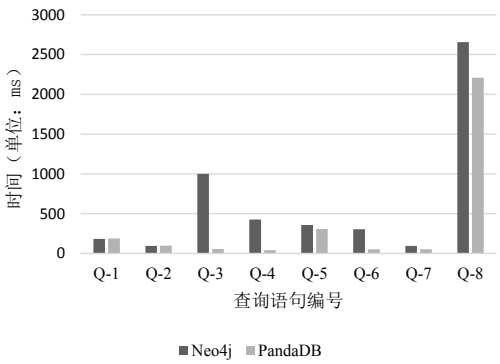

Fig.13 Comparison of query time in co-store

图 13 协存方案查询响应时间对比

从测试结果可看出,由于采用了 ElasticSearch 作为节点属性的协存和索引,在上述查询语句执行测试中,PandaDB 占据明显优势,尤其在节点多属性过滤查询和模糊匹配查询中性能平均提升 2~6 倍.

4.2 分布式性能测试

为验证分布式架构带来的查询响应能力的提升,本文在物理机集群上（物理机配置见表 2）部署了 PandaDB 和 Neo4j 单机版,对并发查询请求的吞吐率进行评估.因 Neo4j 社区版仅支持单机,故其中 PandaDB 部

署在 3 台物理机,Neo4j 部署在一台物理机.

本文选择了图计算分析中的常见查询作为测试用例,如计算节点出度和入度.将超时阈值设定为 300s,在满足 90%查询不超时的前提下,Neo4j 的吞吐量为每秒 15 次查询,PandaDB 的吞吐量为每秒 40 次查询.PandaDB 的吞吐量接近 Neo4j 的 3 倍.

4.3 非结构化数据查询加速测试

为实现对非结构化数据查询的加速,PandaDB 采用了构建语义索引的方式,减少数据传输和调用 AI 服务的次数.本小节基于人脸检测场景,使用 LFW 数据集[11],构建对比实验,通过对比各种方案下的查询耗时,验证语义索引方案的加速效果.

本实验任务目标为:从样本集中找到与目标人脸图片相似度最高的人脸图片.本实验共分四组对比,各组实验内容及实验条件细节如下:

- NOOP:无优化,PandaDB 向 AIPM 发出图片相似度对比请求,逐个将图片以 BLOB 流的形式发送给 AIPM,AIPM 接收请求后,抽取特征,将比对结果返回给 PandaDB.
- DLOC:数据本地化方案,PandaDB 与 AIPM 部署在同一台服务器上,减少 BLOB 传输的网络开销.
- AIIDX:AI 服务缓存方案,PandaDB 向 AIPM 请求数据,AIPM 利用本地缓存的特征数据相应请求.模仿数据流水线工具的实现.
- SEMIDX:语义索引方案,PandaDB 中构建非结构化数据的语义索引（人脸特征向量）,执行查询时,直接调用本地索引数据比较.

Table 5 Comparison of query time under different methods

表 5 不同方法的查询耗时对比

样本数	NOOP	DLOC		AIIDX		SEMIDX	
		耗时(ms)	加速比	耗时(ms)	加速比	耗时(ms)	加速比
1	993	999	0.99	124	8.04	5	199.80
5	1636	1641	1.00	672	2.44	10	164.10
10	3708	3482	1.06	1299	2.68	6	580.33
20	6854	7036	0.97	2448	2.87	7	1005.14
s50	16336	15665	1.04	6169	2.54	8	1958.13
100	32138	30714	1.05	12291	2.50	10	3071.40
500	163154	156025	1.05	61505	2.54	22	7092.05
1000	320760	307251	1.04	122525	2.51	33	9310.64
2000	643462	615047	1.05	244661	2.51	60	10250.78
5000	161698	1542788	1.05	615192	2.51	144	10713.81

表 5 给出了在不同样本数下各方案执行同一个查询的耗时.SEMIDX 方案相比于其他方案减少了数据传输和重复的数据抽取,故理论上该方案在四个方案中速度最快.从表 5 中可以看出在样本数相同的条件下,SEMIDX 方案具有最短的查询时间.在样本数超过 2000 的检索性能对比中,加速比超过一万倍.

4.4 案例:学术图谱实体消歧与可视化

学术图谱泛指以学术内容为主体的领域知识图谱,典型应用如 AMiner[12]和 AceKG[13].本文基于 KDD2020 的参会人员信息[14]构建了一个小型的学术图谱,其中包含论文、作者、机构三类节点,根据论文创作关系和学者与机构间的隶属关系,将作者照片、论文的 PDF 全文等非结构化数据作为属性直接保存在

PandaDB 中.图 14 是 KDD2020 数据的可视化展示1,其中头像属性存储在 PandaDB,且在 PandaDB 的基础上提供了“以图找图”的功能,即实现了基于头像的快速匹配.

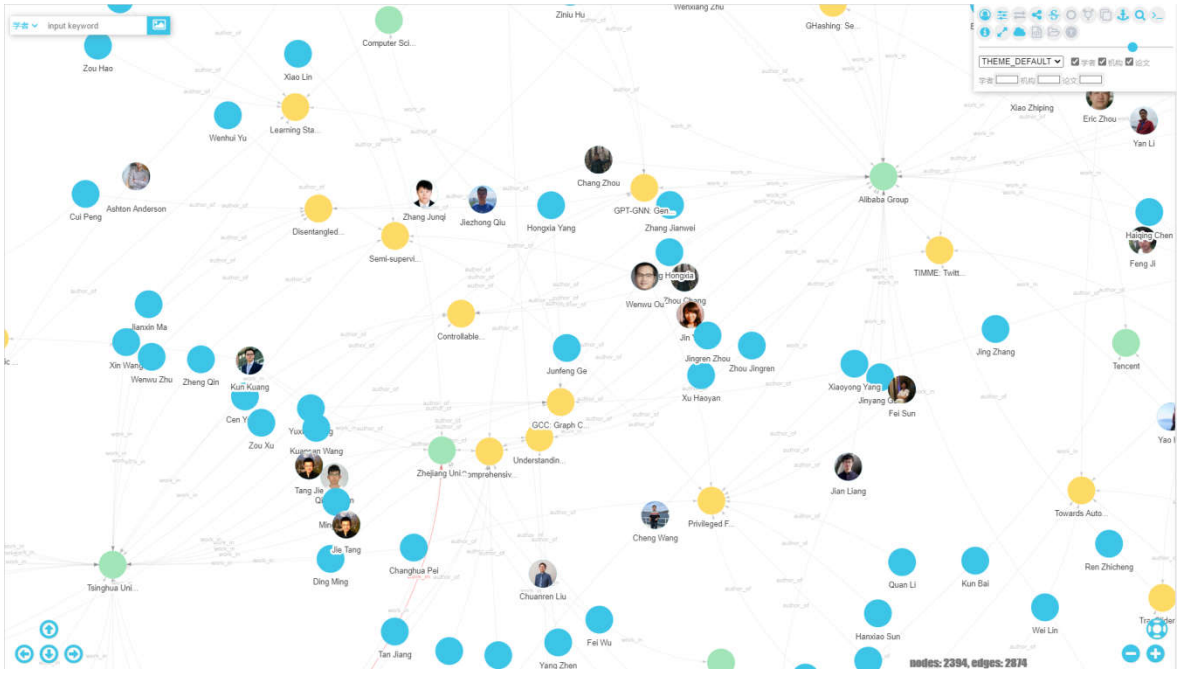

图 14 KDD2020 数据的可视化展示

Figure 14 Visualization of KDD2020

学术图谱原始数据中通常会出现学者重名、不同机构的简称或缩写相同等情况,因此图谱创建过程中通常面临实体消歧问题.目前大多数实体消歧方法都以聚类为基础,文献[15][16][17] 使用表层特征值计算实体间相似度,但这种方法不能充分利用上下文特征.因为基于表层特征的方法面临信息不足的问题,有学者尝试引入外部信息,如 WikiPedia 中的信息,将这些知识资源作为实体的扩展特征,辅助提升聚类准确性,代表工作如[18][19].文献[20][21]使用图计算的方法,充分利用网络数据的结构信息,但并不具有很好的普适性.近年来,随着深度学习技术的巨大进步,有学者利用 embedding 技术[22]和神经网络技术[23]这类方法相比于传统方法具有更好的效果,但准确性方面仍不能满足人们的预期.本案例基于 PandaDB,提出了一种融合结构化属性和非结构化属性的消歧方法.如图 15 所示,某学术图谱中有两个姓名分别为 Park Bill 和 Tom Green 的人物节点以及一个名为 Data Vis 的论文节点,需要判断 T.Green 和 Tom Green 是否为同一实体,从而判断 Park Bill 和 Tom Green 之间是否存在合作关系.由于论文中存在的属性数据不足,消歧很有困难.基于 PandaDB 对非结构化信息抽取及语义比较能力,可以充分利用 Tom Green 的照片以及论文作者照片列表,从而达到消歧目的.图 15 下半部分给出了完成该操作的 CypherPlus 语句,其中<:操作符表示包含关系,只需要找到对应的节点,判断 n.photo<:p.screenshot 成立与否,即可计算出二者之间是否存在合作关系.

1 基于开源项目 InteractiveGraph:https://github.com/grapheco/InteractiveGraph

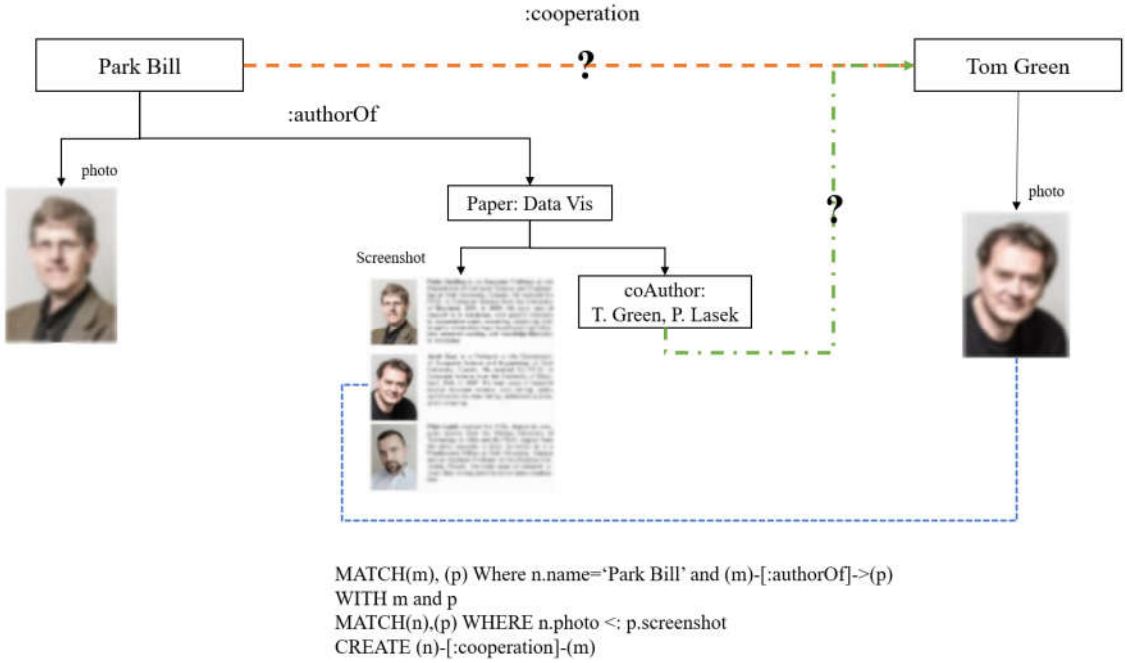

Fig.15 Diagram of Entity Disambiguation based on PandaDB

图 15 基于 PandaDB 实现实体消歧示意图

5 相关工作

为实现数据应用中的多元异构数据的统一管理和分析,一种方法是通过 ETL 工具,采用某种统一的形式处理异构数据.但这种方法成本较高,而且数据和分析需求的变化会使原本的 ETL 流程失效[24]. 异构数据管理可以支持对多模型数据的存储和统一查询.学术界在上世纪 80 年代就提出了联邦数据库的解决概念,Multibase[25]是其中的代表系统,这类系统的特点是定义全局 schema 和基于映射的查询语言,用户基于全局 schema 查询,系统将查询映射到分区 schema.另一类以 Spark SQL[26]为代表的系统提供统一的 API,允许用户以关系模型使用数据,在提升效率和简化查询方面有较好的效果.BigIntegator[27],Forward[28],D4M[29]等系统整合了 NoSQL,其中 Forward 以基于 json 的数据模型组织数据,D4M 使用关联阵列(associative-array)数据模型,这种较为灵活的模型使系统可以在异构数据上执行查询.

在数据管理领域,属性图模型[30]是一种常用的管理图数据的数据模型,属性图中的节点和关系都可以被赋予标签和关联任意键值对形式的属性[31].属性图增加了节点和边的信息,同时又没有改变图的整体结构.目前,属性图模型被图数据库业界广泛采用[32][33],包括著名的图数据库 Neo4j[34]、Titan[35]等.Neo4j 是目前应用较为广泛的一款开源图数据库,其具有从原生图数据存储到可视化插件,再到图数据分析插件的丰富生态.JanusGraph[36]是在 Titan 基础上开发的一种基于属性图的分布式图数据库.JanusGraph 采用存储层和查询引擎分离的设计,可以使用 Cassandra 或 HBase 作为存储层.JanusGraph 通过使用第三方分布式索引库 ElasticSearch、Solr 和 Lucene 实现检索功能.其他的图数据库还包括:Amazon 的 Neptune[37]、微软的 Azure CosmosDB[38]、TigerGraph[39]、OrientDB[40]等.

人工智能技术已经广泛的应用到图像识别、语音识别、机器翻译等领域中.为了实现对语音、图片等多种信息的理解,多模态机器学习的研究尝试在机器学习的角度整合模型对非结构化数据的理解能力[41][42].人工智能与数据库的交叉研究一直是学术界研究的热点内容.人工智能与数据库的融合存在 AI4DB（AI for

Database）和 DB4AI（Database for AI）两个方向[43].AI4DB 旨在通过 AI 技术提高 DB 的效率和能力,如自动化数据库参数调优[44] [45]、基数估计[46]、索引推荐[47]、查询优化[48][49]等.DB4AI,是以数据库为 AI 算法提供数据服务,供算法进行训练和学习.例如,借助数据库的统一 SQL 接口,为用户提供自定义的函数协助构建模型；通过数据库张量计算协助模型训练；通过持久化 AI 模型以重复使用.

随着大数据时代的来临,海量数据的存储与计算使得单机服务已经无法满足需求,越来越多的任务需要分布式系统支持.保证分布式系统的可靠性和一致性至关重要.解决这个问题的一个著名算法是由 Lamport 提出的 Paxos 算法[50][51].此后为了适应不同的工程环境,研究人员在 Paxos 的基础上提出了很多新的算法[52].比较著名的有 Multi-Paxos[51][53]、Liskov 等人提出的 VR (viewstamped replication)算法[54][55]、雅虎公司设计的 ZAB(Zookeeper's atomic broadcast)算法[56]、Ongaro 等人提出的 Raft 算法[57]等.

6 总结与展望

本文从结构化/非结构化数据的融合管理和即席查询需求角度出发,分析了目前多元异构数据融合管理方面统一表示和交互式查询等难点.在此基础上提出具备对异构数据实现统一表示能力的属性图扩展模型,并提出针对在线查询和计算的属性操作符与查询语法.在第 3 节,本文提出基于智能属性图模型的分布式数据融合管理系统 PandaDB,该系统实现了结构化/非结构化数据的高效存储管理,并提供了灵活的 AI 算子扩展机制,具备对多元异构数据内在信息的即席查询能力.测试实验和案例证明,PandaDB 在大规模属性过滤查询和高并发查询响应上具备较好的性能表现.同时 PandaDB 可应用在学术图谱实体消歧与可视化等多元异构数据融合管理的场景.

目前 PandaDB 还存在着一些不足,一方面,AIPM 模块与系统相对独立部署,这种模式降低了系统的耦合性,有利于扩展和维护,但面对大规模的非结构化数据信息查询请求时,模块间信息传输的开销较大.未来应研究更为合理的 AI 功能集成机制,结合多元异构数据即席查询的场景特性设计任务调度方法,提升系统性能.另一方面,从智能属性中抽取内嵌式属性的操作,目前还缺乏有效的缓存和预测机制,造成即席抽取的过程延时较大.PandaDB 将进一步结合应用,进一步提升系统的性能和稳定性,从而提升多元异构数据融合管理的能力.

References:

[1] Gantz J, Reinsel D. Extracting value from chaos. IDC iview, 2011, 1142(2011): 1-12.

[2] Liu, Ye, et al. MMKG: Multi-Modal Knowledge Graphs. European Semantic Web Conference, 2019, pp. 459–474.

[3] Buneman P, Davidson S, Fernandez M, et al. Adding structure to unstructured data. International Conference on Database Theory. Springer, Berlin, Heidelberg, 1997: 336-350.

[4] Li W, Lang B. A tetrahedral data model for unstructured data management. Sci China Inf Sci, 2010, 53: 1497–1510, doi: 10.1007/s11432-010-4030-9

[5] Gerber D, Hellmann S, Bühmann L, et al. Real-time RDF extraction from unstructured data streams. International semantic web conference. Springer, Berlin, Heidelberg, 2013: 135-150.

[6] Sears R, Van Ingen C, Gray J. To blob or not to blob: Large object storage in a database or a filesystem?. arXiv preprint cs/0701168, 2007.

[7] Zhu Y, Du N, Tian H, et al. LaUD-MS: an extensible system for unstructured data management. 2010 12th International Asia-Pacific Web Conference. IEEE, 2010: 435-440.

[8] Zhang Xiao et al. Managing a large shared bank of data by using Free-Table . Proceedings of the 12th Asia-Pacific Web Conference(APWeb 2010), Busan, Korea, Apr 6-8, 2010: 441-446.

[9] Zhou NN, Zhang X, et al. Design and Implementation of Adaptive Storage Management System in MyBUD . Journal of Frontiers of Computer Science & Technology, 2012,6(08): 673-683 (in Chinese with English abstract).

[10] Francis, Nadime, et al. Cypher: An Evolving Query Language for Property Graphs. Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 1433–1445.

[11] Huang G B, Mattar M, Berg T, et al. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments[C]. 2008.

[12] Wan H, Zhang Y, Zhang J, et al. Aminer: Search and mining of academic social networks. Data Intelligence, 2019, 1(1): 58-76.

[13] Wang R, Yan Y, Wang J, et al. Acekg: A large-scale knowledge graph for academic data mining. Proceedings of the 27th ACM international conference on information and knowledge management. 2018: 1487-1490.

[14] KDD2020[WB/OL]. (2020-9-21) https://www.aminer.cn/conf/kdd2020/homepage

[15] Bagga A, Baldwin B. Entity-Based Cross-Document Core f erencing Using the Vector Space Model. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1. 1998: 79-85.

[16] Pedersen T, Purandare A, Kulkarni A. Name discrimination by clustering similar contexts. International Conference on Intelligent Text Processing and Computational Linguistics. Springer, Berlin, Heidelberg, 2005: 226-237.

[17] Chen Y, Martin J H. Towards robust unsupervised personal name disambiguation. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). 2007: 190-198.

[18] Cucerzan S. Large-scale named entity disambiguation based on Wikipedia data. Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL). 2007: 708-716.

[19] Han X, Zhao J. Named entity disambiguation by leveraging wikipedia semantic knowledge. Proceedings of the 18th ACM conference on Information and knowledge management. 2009: 215-224.

[20] Hassell J, Aleman-Meza B, Arpinar I B. Ontology-driven automatic entity disambiguation in unstructured text. International Semantic Web Conference. Springer, Berlin, Heidelberg, 2006: 44-57.

[21] Minkov E, Cohen W W, Ng A Y. Contextual search and name disambiguation in email using graphs. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. 2006: 27-34.

[22] Zhang B, Al Hasan M. Name disambiguation in anonymized graphs using network embedding. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017: 1239-1248.

[23] Huang H, Heck L, Ji H. Leveraging deep neural networks and knowledge graphs for entity disambiguation. arXiv preprint arXiv:1504.07678, 2015.

[24] Tan R, Chirkova R, Gadepally V, et al. Enabling query processing across heterogeneous data models: A survey. 2017 IEEE International Conference on Big Data (Big Data). IEEE, 2017: 3211-3220.

[25] Smith J M, Bernstein P A, Dayal U, et al. Multibase: integrating heterogeneous distributed database systems. Proceedings of the May 4-7, 1981, national computer conference. 1981: 487-499.

[26] Armbrust M, Xin R S, Lian C, et al. Spark sql: Relational data processing in spark. Proceedings of the 2015 ACM SIGMOD international conference on management of data. 2015: 1383-1394.

[27] Zhu M, Risch T. Querying combined cloud-based and relational databases. 2011 International Conference on Cloud and Service Computing. IEEE, 2011: 330-335.

[28] Ong K W, Papakonstantinou Y, Vernoux R. The SQL++ unifying semi-structured query language, and an expressiveness benchmark of SQL-on-Hadoop, NoSQL and NewSQL databases. CoRR, abs/1405.3631, 2014.

[29] Kepner J, Arcand W, Bergeron W, et al. Dynamic distributed dimensional data model (D4M) database and computation system. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2012: 5349-5352.

[30] Ehrig H , Prange U , Taentzer G . Fundamental theory for typed attributed graph transformation.. Lecture Notes in Computer ence, 2004, 3256:161-177.

[31] Robinson I, Webber J, Eifrem E. Graph databases: new opportunities for connected data[M]. " O'Reilly Media, Inc.", 2015.

[32] Wang X, Zou L, Wang CK, Peng P, Feng ZY. Research on Knowledge Graph Data Management: A Survey.Ruan Jian Xue Bao/Journal of Software, 2019,30(07):2139-2174 (in Chinese with English abstract).

[33] Angles R. A comparison of current graph database models. 2012 IEEE 28th International Conference on Data Engineering Workshops. IEEE, 2012: 171-177.

[34] The Neo4j Team. The Neo4j Manual v3.4. 2018. https://neo4j.com/docs/developer-manual/current/

[35] Spmallette. Titan—Distributed graph database. 2018. http://titan.thinkaurelius.com/

[36] JanusGraph Authors. JanusGraph—Distributed graph database. 2018. http://janusgraph.org/

[37] Amazon Web Services, Inc. Amazon Neptune—Fast, reliable graph database build for cloud. 2018. https://aws.amazon.com/neptune/

[38] Microsoft Azure. Microsoft Azure Cosmos DB. 2018. https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

[39] TigerGraph. TigerGraph—The first native parallel graph. 2018. https://www.tigergraph.com/

[40] Callidus Software Inc. OrientDB-multi-model database. 2018. http://orientdb.com/

[41] Baltrušaitis T, Ahuja C, Morency L P. Multimodal machine learning: A survey and taxonomy. IEEE transactions on pattern analysis and machine intelligence, 2018, 41(2): 423-443.

[42] Ramachandram D, Taylor G W. Deep multimodal learning: A survey on recent advances and trends. IEEE Signal Processing Magazine, 2017, 34(6): 96-108.

[43] Li GL, Zhou XH. XuanYuan: An AI-native database systems. Ruan Jian Xue Bao/Journal of Software, 2020, 31(3):831−844 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5899.htm

[44] hang J, Liu Y, Zhou K, Li G. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: Proc. of the SIGMOD. 2019.

[45] Wang W, Zhang M, Chen G, Jagadish HV, Ooi BC, Tan K. Database meets deep learning: Challenges and opportunities. SIGMOD Record, 2016,45(2):1722.

[46] Kipf A, Kipf T, Radke B, Leis V, Boncz PA, Kemper A. Learned cardinalities: Estimating correlated joins with deep learning. In: Proc. of the CIDR. 2019.

[47] Pedrozo WG, Nievola JC, Ribeiro DC. An adaptive approach for index tuning with learning classifier systems on hybrid storage environments. In: Proc. of the HAIS. 2018. 716729.

[48] Krishnan S, Yang Z, Goldberg K, Hellerstein JM, Stoica I. Learning to optimize join queries with deep reinforcement learning. CoRR, abs/1808.03196, 2018.

[49] Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration. In: Proc. of the 1st Int'l Workshop on Exploiting Artificial Intelligence Techniques for Data Management. 2018. 3:13:4.

[50] Lamport L.The part-time parliament.ACM Transactions on Computer Systems, 1998, 16 (2) :133-169

[51] Lamport L.Paxos made simple.ACM SIGACT News, 2001, 32 (4) :18-25

[52] Wang J, Zhang MX, Wu YW, Chen K, Zheng WM. Paxos-like Consensus Algorithms: A Review. Journal of Computer Research and Development, 2019, 56(04): 692-707 (in Chinese with English abstract).

[53] Chandra T D, Griesemer D, Redstone J. Paxos made live: An engineering perspective. Proc of the 26th Annual ACM Symp on Principles of Distributed Computing (PODC'07). New York: ACM, 2007:398-407

[54] Oki B M, Liskov B H. Viewstamped replication: A new primary copy method to support highly-available distributed systems. Proceedings of the seventh annual ACM Symposium on Principles of distributed computing. 1988: 8-17.

[55] Liskov, Barbara, and James Cowling. Viewstamped Replication Revisited. 2012.

[56] Medeiros A. ZooKeeper's atomic broadcast protocol: Theory and practice[R]. Technical report, 2012.

[57] Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14). 2014: 305-319.

附中文参考文献:

[3] 李未, 郎波. 一种非结构化数据库的四面体数据模型. 中国科学: 信息科学, 2010, 40(8): 1039-1053.

[8] 周宁南,张孝,孙新云,琚星星,刘奎呈,杜小勇,王珊.MyBUD 自适应分布式存储管理的设计与实现.计算机科学与探索,2012,6(8):673-683.

[30] 王鑫,邹磊,王朝坤,彭鹏,冯志勇.知识图谱数据管理研究综述.软件学报,2019,30(7):2139-2174.

[39] 李国良,周煊赫.轩辕:AI 原生数据库系统.软件学报,2020,31(3):831-844.

[50] 王江,章明星,武永卫,陈康,郑纬民.类 Paxos 共识算法研究进展.计算机研究与发展,2019,56(4):692-707.