计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(9):53−61 [doi: 10.15888/j.cnki.csa.008139]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于查询语言转换的多源数据统一访问框架①

李跃鹏[1,2], 温亮明[1,2], 黎建辉[1]

[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: 黎建辉, E-mail: lijh@cnic.cn

摘　要: 大数据时代, 需要集成多样化数据管理分析工具完成业务需求, 然而不同工具的数据访问接口标准不一, 需要借助适配器进行接口转换以实现数据管理分析工具协作. 因此, 如何快速集成不同工具成为学术界与工业界亟待解决的问题. 本文提出了一种基于查询语言转换的多源数据统一访问框架 BAF4DUA (Bi-Adapter Framework for Data Unified Access), 该框架采用双端接口适配方式, 在数据提供者工具端和数据消费者工具端分别引入适配器对系统数据与查询语义进行适配, 将查询语言与数据存储模型相分离, 从而实现了数据提供者与数据消费者之间多对多、即插即用的数据访问, 提高了应用的灵活性与扩展性.

关键词: 统一查询; 多源异构数据; 双适配器; 中间模型; 语言转换

引用格式: 李跃鹏,温亮明,黎建辉.基于查询语言转换的多源数据统一访问框架.计算机系统应用,2021,30(9):53–61. http://www.c-s-a.org.cn/1003-3254/8139.html

Framework for Heterogeneous Data Unified Access Based on Language Translating

LI Yue-Peng[1,2], WEN Liang-Ming[1,2], LI Jian-Hui[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In an era of big data, application needs to integrate various data management tools to fulfill its business requirements. However, these tools differ in their APIs and need adapters to communicate with each other. Therefore, how to integrate data management tools quickly becomes an important research problem in academia and industry. This study introduces a language translating based framework for heterogeneous data unified access, which is called the Bi-Adapter Framework for Data Unified Access (BAF4DUA). In BAF4DUA, two types of adapters are located at the data provider endpoint and data consumer endpoint to translate the query semantics and data model. The query language and data model are decoupled by the two adapters, thereby enabling many-to-many and plug-and-play data access modes between data providers and data consumers. In addition, the decoupled framework can also promote the flexibility and scalability of the application system.

Key words: unified access; heterogeneous data; bi-adapter; middle data model; language translating

1　引言

大数据应用系统是由多种数据处理工具构成的集成性系统, 其数据结构、数据存储位置、数据管理与分析工具等均具有多样化特点[1,2]. 例如在电商系统中, 通过关系数据库管理用户交易数据, 商品数据存储于文档数据库; 为加快系统响应速度, 通过 Key-Value 数

① 基金项目: 国家重点研发计划 (2016YFB1000600); 中国科学院战略性先导科技专项 (A 类) 子课题 (XDA19020104)
Foundation item: National Key Research and Development Program of China (2016YFB1000600); Category A Strategic Priority Research Program of Chinese Academy of Sciences (XDA19020104)
收稿时间: 2020-12-22; 修改时间: 2021-01-25; 采用时间: 2021-02-26; csa 在线出版时间: 2021-09-02

© 中国科学院软件研究所 http://www.c-s-a.org.cn

据库缓存高频率访问数据; 统计报表使用流处理工具、SQL 查询引擎等进行历史数据统计分析. 在科学大数据管理领域, 科研项目通过领域关系型数据库管理从传感器、遥感卫星等科学装置获得的实验数据; 使用领域非结构化数据库进行高能物理对撞机事件数据的存储与索引; 通过图数据库管理物种、基因组等知识图谱数据; 借助领域数据分析工具对各类实验数据进行统计与交叉分析, 进而得出实验结论[3]. 大数据集成系统使用的数据处理工具分为两类: 数据提供者使用标准查询语言、WebService 等接口方式提供数据库、开放平台等基础数据管理服务; 数据消费者调用多个数据提供者接口获取数据, 并通过标准查询语言、WebService 等接口对多源异构数据进行查询、统计、分析、展示等操作, 如 SQL 查询引擎、数据仓库、管理系统等. 数据消费者与数据提供者之间的数据访问方式具有以下特点:

(1) 数据访问接口各异: 数据提供者接口需经转换才能符合数据消费者的接口格式要求. 比如报表系统中, 数据消费者使用 SQL 语言对数据进行统计, 然而数据提供者的服务接口为文档查询、WebService 以及文件等形式.

(2) 数据访问模式多样: 数据消费者与数据提供者之间是多对多数据访问方式. 例如数据仓库使用 SQL 语言对多个异构数据源进行分析; 而异构数据源除了为数据仓库提供服务, 还会被多个部门或外部应用系统消费.

(3) 松耦合方式集成: 数据消费者与数据提供者独立运行、部署与迭代更新. 例如作为数据提供者, 存储用户、业务与实验等数据的数据库会根据技术、场景等因素进行更新, 然而数据消费者的访问接口却并不改变; 相反, 数据消费者会根据业务、性能等需要调整数据消费者的数据访问接口.

基于以上数据访问特点, 要实现数据提供者接口与消费者接口的交互协作, 集成系统必须提供一个接口适配器以满足双方需要, 目前已有研究特定应用场景接口适配器的相关工作. 随着数据提供者与数据消费者工具的更新变化, 这种针对特定应用场景的一对一型接口适配器势必难以适应数据迁移与工具迭代更新需要, 因此我们提出了一种基于查询语言转换的多源数据统一访问框架, 通过引入中间数据模型和双端适配器将数据访问接口 API 与实际数据模型相分离, 使得集成系统具有更好的灵活性和扩展性.

2 相关工作

接口适配器以数据消费者数据模型或数据提供者数据模型作为中间数据模型. 首先将数据提供者或数据消费者的 Schema 映射为中间数据模型, 然后通过数据转换或查询语言转换方式完成接口适配转换. 数据转换适配器将数据提供者所持有数据动态或一次性转换为消费者可识别的数据格式进行存储或处理; 查询语言转换适配器将数据消费者端的查询语言转换为数据提供者端的查询语言.

根据数据消费者对数据提供者的访问方式, 数据消费者与数据提供者的接口适配方式分为一对一适配、多对一适配以及多对多适配 3 种 (如图 1(a) 至图 1(c) 所示).

(1) 一对一适配: 采用查询语言转换方式将数据消费者接口转换为数据提供者查询语言. 例如文献 Sql2Saprql (RETRO)[4]、Sparql2Sql (D2RQ)[5]、Cypher2sql (Cytosm)[6] 等提出了将不同查询语言转换为 SQL 查询语句或将 SQL 查询语句转换为其它查询语言的方法.

(2) 多对一适配: 以消费者数据模型为中间数据模型, 数据提供者向统一模型动态提供适配数据. 例如 SparkSQL[7], Presto[8]、Impala[9] 等 SQL 查询引擎适配器将关系数据库、文档数据库、CSV 文件、JSON 文件等数据源动态转换为临时关系表来实现 SQL 统一查询. Gradoop 适配器将异构数据动态转换为属性图模型的节点和边来实现 Cypher 语言统一查询[10].

(3) 多对多适配: 为数据提供者和数据消费者提供接口适配. 例如 Polystore[11] 要求系统支持多种全局数据模型与本地数据模型之间的转换, 通过数据提供者与数据消费之间的成对查询语言转换适配器为上层应用提供多样化数据查询接口.

以数据访问接口适配方法存在的问题是中间数据模型与查询语言模型绑定, 使得接口适配器只在单个数据提供者与数据消费者之间产生效用. 对于数据转换适配器, 当多个数据消费者的数据模型不同时, 数据提供者必须向数据消费者提供不同的适配数据. 对于查询语言转换适配器, 多对多访问同样必须采用 case by case 方式实现成对查询语言转换. 当系统中存在 n 个异构数据提供者和 n 个数据消费者时, 要支持多对多访问就必须

© 中国科学院软件研究所 http://www.c-s-a.org.cn

开发 $n!$个适配器, 如果新增一个异构数据提供者则需要实现 $2n$ 个转换器, 从而影响系统对新数据源的扩展性. 因此, 本文尝试探索提出一种双适配器数据统一访问框架 (如图 1(d) 所示), 通过数据提供者与数据消费者两端适配器将查询语言与数据模型分离, 从而对新数据提供者与数据消费者的支持只要按需实现一个适配器即可.

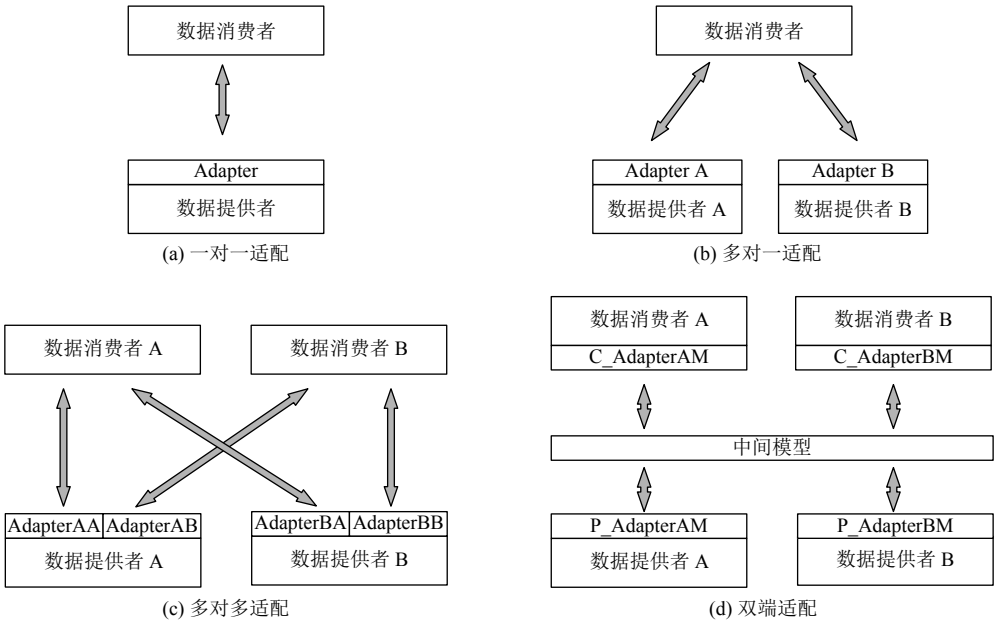

图 1 数据提供者与消费者接口适配方式

3 框架设计

双 Adapter 数据统一访问框架 BAF4DUA 将多个异构数据消费者与数据提供者接口进行适配, 为上层应用提供多样化的数据服务接口. 该框架使用中间模型统一表示数据提供者所持有数据, 中间查询计划统一表示消费者与提供者查询语言的语义; 数据消费者与数据提供者两端分别与中间模型进行接口适配, 将数据访问接口与数据模型分离, 从而支持消费者与提供者多对多数据访问, 提高系统的灵活性与扩展性.

如图 2 所示, 双 Adapter 数据统一访问框架由数据消费端 Adapter、中间数据模型、中间查询计划、数据提供者端 Adapter 以及数据映射 5 部分构成.

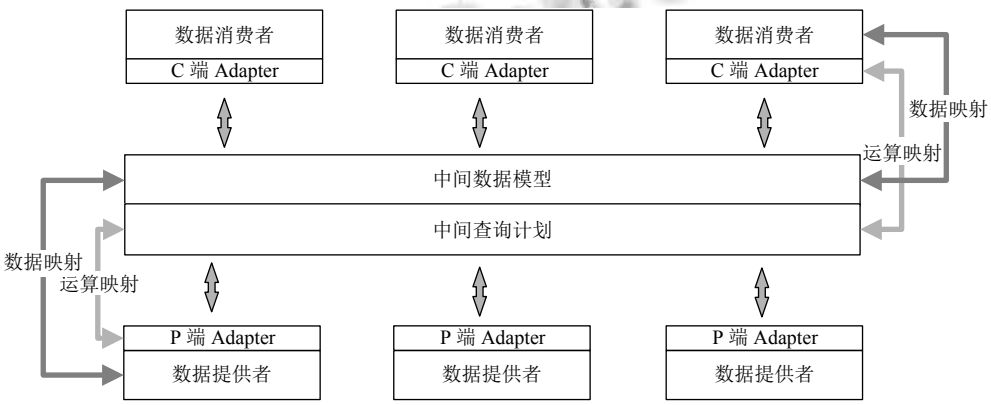

图 2 双 Adapter 数据统一访问框架

(1) 数据消费者端 Adapter 是系统数据访问请求的编码器. 它将消费者查询语句涉及的目标数据与中间数据模型进行映射, 同时将查询语句的语义编码为中间查询计划. 根据系统设计需要, 数据消费者端 Adapter 还需对查询结果进行格式转换以满足消费者应用需求.

(2) 数据提供者端 Adapter 是系统数据访问请求的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

解码器, 它将中间查询计划解码为数据提供者可识别的查询语句, 实现中间数据模型与数据提供者的数据映射. 根据系统设计需要, 数据提供者端 Adapter 还需要将查询结果转换统一数据格式.

(3) 中间数据模型为系统多源异构数据提供统一语义视图, 它定义数据的统一表示方式, 记录中间模型的元素构成与数据持有者的映射关系, 支持数据的增、删、改、查、合并等操作. 根据系统设计需要, 中间数据模型还需定义通用数据格式以供数据消费者与数据提供者之间进行查询结果转换.

(4) 中间查询计划将数据提供者和数据消费者接口对数据的操作进行抽象化处理, 定义系统支持的运算类型, 实现数据提供者与数据消费者查询操作与中间查询计划运算的相互转换. 由于不同数据提供者和数据消费者对数据运算操作的支持程度不同, 根据系统设计需要, 还必须对不支持的数据操作进行补偿运算.

(5) 数据映射将数据提供者所持有数据与中间数据模型进行映射, 在映射过程中数据提供者根据中间数据模型的定义, 将其所持有的数据按照映射规范向中间数据模型注册. 在此过程中, 数据映射规范需要适量增加或减少原有数据的限制以满足中间数据模型的定义.

4 框架实现

为了验证 BAF4DUA 框架在实际系统中的应用, 本文实现了一个多对多数据统一访问系统, 其中数据消费者接口为标准查询语言, 数据提供者接口为常用数据库. 该系统中间数据模型使用实体与关系对数据进行统一表述, 数据映射将数据库、WebService 等查询接口的数据对象映射为中间数据模型元素. 中间查询计划选择四类运算形式化描述查询语句的语义信息, 消费者与提供者 Adapter 将各自接口与中间数据模型和中间查询计划进行适配. 目前系统支持通过 SQL、Cypher、MongoDB 等查询语言接口对不同数据提供者进行数据访问, 在不影响系统其它组成部分的情况下, 可实现数据库迁移、缓存、统一管理等应用需求.

4.1 中间数据模型与数据映射

中间数据模型对集成系统中的异构数据采用统一方式进行语义描述, 系统数据用实体 (Entity) 与关系 (Relation) 两个概念对常见数据源进行描述[12]. 实体为具有相似意义的数据集合, 实体元素的具体数据格式不限, 既包括结构化形式的表数据, 也包括非结构化形式的文档数据; 关系为实体间的语义关联, 可通过多种方式进行表示, 关系表示形式包括实体条件表达式或存储的关系数据. 中间数据模型的形式化表示如下:

$$UM = \langle E, R, S, L, K, \rho \rangle$$

其中, E 表示实体集合, $R \in (E \times E)$ 表示关系集合, S 表示数据源集合, L 表示用于实体与关系标识的标签集合, K 表示 L、S 与 E、R 之间的映射关系, ρ 表示 S 与 R 之间的数据映射方式.

异构数据源 S 包括两种形式: 一是以标准查询语言为访问接口的数据库管理工具, 如关系数据库、文档数据库以及图数据库等; 二是其它非标准化数据访问接口, 如 WebService、Shell 命令以及文件系统等. S 中的每个元素必须包含连接访问该数据源的所有参数, 如连接字符串、URL 以及用户名密码等.

映射关系 K 由两部分构成: 标签映射 K_l:$(E \cup R) \rightarrow L$ 表示标签与 ER 元素之间的映射关系, 即实体集合与关系由一个标签唯一标识; 数据映射 K_s:$(E \cup R) \rightarrow (S \cup \rho)$ 表示数据源与 Entity、Relation 元素之间的映射关系, 即 Entity 与 Relation 由数据源与数据映射方式构成.

针对不同数据源, 数据映射方式 ρ 采用多种方法将异构数据源映射为 Entity 和 Relation 元素. 对于提供标准查询语言接口的数据源, 可使用通用的数据映射规范自动将数据源映射到中间数据模型. 如表 1 所示, 将关系模型数据中的 Table 映射为中间数据模型的 Entity 元素, 将“外键”或属性相关的逻辑表达式映射为中间数据模型中的 Relation 元素; 将文档模型数据源中的 Collection 映射为中间数据模型的 Entity, 将文档的 DBref 或属性的逻辑表达式映射为中间数据模型的 Relation; 将属性图模型数据源中具有相同 Label 的节点集合映射为 Entity, 而将具有同样 Label 的边集合映射为 Relation.

表 1 数据映射关系

中间模型	关系模型	文档模型	属性图模型	Web服务等
实体	表	文档集合	Label节点集合	服务
属性	列	属性	属性	自定义
关系	外键或表达式	DBref或表达式	Label 边集合	自定义

对于无标准查询语言接口的数据提供者, 需要根据数据提供者的数据访问接口特点自定义数据源与中间数据模型的映射方式. 例如 WebService 接口与中间数据模型的映射需要根据服务 URL、Token 以及业务

© 中国科学院软件研究所 http://www.c-s-a.org.cn

参数等与实体、关系进行映射.

4.2 中间查询计划

查询计划是对查询语句所表达语义的形式化描述, 它是一个节点为数据模型运算的树形结构. 中间查询计划 p:$um_{src}\rightarrow um_{target}$ 描述了从数据 um_{src} 向查询目标数据 um_{target} 转换的运算过程, 它代表了查询语句的查询目标, 系统支持的运算类型越多, 执行计划能够支持的查询功能越丰富. 在实现过程中, 我们选择了 4 类中间数据模型运算作为中间查询计划的基础组成部分:

(1) *scan*(um_{src}, um_{target}, *item*) 运算对参加运算的中间数据模型中实体与关系进行选择. 其中 *item* 指 um_{src} 中的实体或关系元素, *scan* 运算将 um_{src} 中的 *item* 元素加入 um_{target}. 如果 *item* 类型为 Relation, *scan* 运算同时将与该 Relation 关联的 Entity 添加到 um_{target}.

(2) *filter*(um_{target}, *item*, *cond*, *type*) 运算对 um_{target} 中 Entity 或 Relation 集合内数据对象进行条件约束. 其中 *item* 指 um_{target} 中的实体或关系元素, *cond* 为 *item* 中数据对象满足的逻辑表达式条件, *type* 为过滤运算类型. 根据不同过滤运算类型, *filter* 运算采用不同形式过滤 um_{target} 中的数据.

(3) *link*(um_{src}, um_{target}, $item_1$, $item_2$) *运算*在 Entity 元素 $item_1$ 与 $item_2$ 之间建立一个关系加入 um_{target} 中. 建立 Relation 的方式有两种: 一是通过 $item_1$ 与 $item_2$ 之间的逻辑表达式创建一个临时 Relation; 二是从 um_{src} 选择与 $item_1$、$item_2$ 对应的 Relation 加入 um_{target}.

(4) *project*(um_{target}, *cond*) *运算*根据 *cond* 条件选择 um_{target} 中的元素或元素内容保留参与后续运算. *cond* 条件包括两类: um_{target} 中 Entity 或 Relation 元素; Entity 或 Relation 数据的属性、列等子元素.

4.3 消费者端 Adapter 与数据提供者端 Adapter

BAF4DUA 架构中适配器对数据消费者和数据提供者的数据查询接口进行转换翻译: 第 1 步, 数据消费者端 Adapter 将数据消费者查询语句编码为中间查询计划统一表示; 第 2 步, 数据提供者端 Adapter 将中间查询计划解码为数据提供者可识别的查询语句. 通过中间查询计划对查询语句的统一形式化表示, BAF4DUA 框架将查询语言接口与数据模型解耦.

查询语句在数据源执行过程中会解析节点为数据模型运算的逻辑查询计划树, 数据源执行引擎遍历查询计划树依序执行运算节点. 数据消费者端 Adapter 对查询语句的语义编码过程需实现中间查询计划与逻辑查询计划、中间数据模型与数据源的运算映射两部分内容:

(1) 数据映射: 主要过程是解析数据消费者请求的查询语句, 获取查询语句的目标数据对象, 根据数据映射方式 ρ 获取目标数据对象与中间数据模型 Entity 和 Relation 的对应关系, 收集相应的数据作为后续运算参数.

(2) 运算映射: 主要过程是将数据消费者查询语句解析为逻辑查询计划树并自底向上遍历, 根据根节点到当前运算节点路径计算逻辑查询计划树中的每个运算节点所处的状态并据此确定后续运算映射. 逻辑查询计划树的运算节点与中间执行计划运算节点之间存在一对一、一对多、多对一以及多对多关系, 根据运算映射关系生成中间执行计划的运算节点以及中间执行计划树.

与数据消费者端对查询语句的编码过程相反, 数据提供者端 Adapter 对中间执行计划树解码操作, 生成数据提供者查询语言接口的逻辑查询计划树. 解码过程根据数据映射方式 ρ 从中间执行计划运算节点的参数中抽取出数据源的操作对象, 实现中间数据模型与数据源的有效映射. 根据中间执行计划运算节点与逻辑查询计划运算节点的对应关系生成逻辑查询计划树, 进而生成目标查询语句提交给数据提供者执行.

5 案例示范

5.1 BAF4DUA 案例工具实现

根据第 4 节提出的 BAF4DUA 框架实现方案, 本文实现了一个支持关系模型、属性图模型和文档模型与中间数据模型的相互映射, 以及 SQL、Cypher、Mongo 命令 3 种查询语言的相互转换的案例工具.

如图 3 所示, BAF4DUA 案例工具的具体实现主要包括 10 个类. UnifiedDataModel 类为 Entity 与 Relation 组成的图数据结构, 在使用 BAF4DUA 案例工具之前需根据数据提供者的数据进行初始化, 通过 UnifiedDataModel 类, 查询语言转换过程中可以确定中间查询计划、数据提供者查询对象、数据消费者查询对象之间的映射关系. 接口类 Encoder 与 Decoder 包含查询语言与中间查询计划 UnifiedQueryPlan 进行转换的 *encode* 与 *decode* 函数. SqlEncoder、CypherEncoder 以及 MongoEncoder 为 Encoder 接口类的具体实现, 其中 SqlEncoder 首先利用 Apache Calcite[13] 将 SQL 语句

© 中国科学院软件研究所 http://www.c-s-a.org.cn

解析为 SQL 查询计划, 然后通过遍历 SQL 查询计划将 SQL 查询计划中的 JOIN、FILTER 运算以及表达式转换为 UnifiedQueryPlan 中对应运算 Operator. 类似的, 我们使用开源工具 Cytosm[14] 以及 Mongo-Shell-Like-Query[15] 分别将 Cypher 与 Mongo 命令分别解析成相应的查询计划, 根据 4.3 节所述遍历该查询计划生成对应的 UnifiedQueryPlan 对象. SqlDecoder、CypherDecoder 以及 MongoDecoder 为 Decoder 接口类的具体实现. 其中 SqlDecoder 假设最终生成的 SQL 语句模式为:

SELECT <RESULT>
FROM <TABLES>
WHERE <CONDITION>

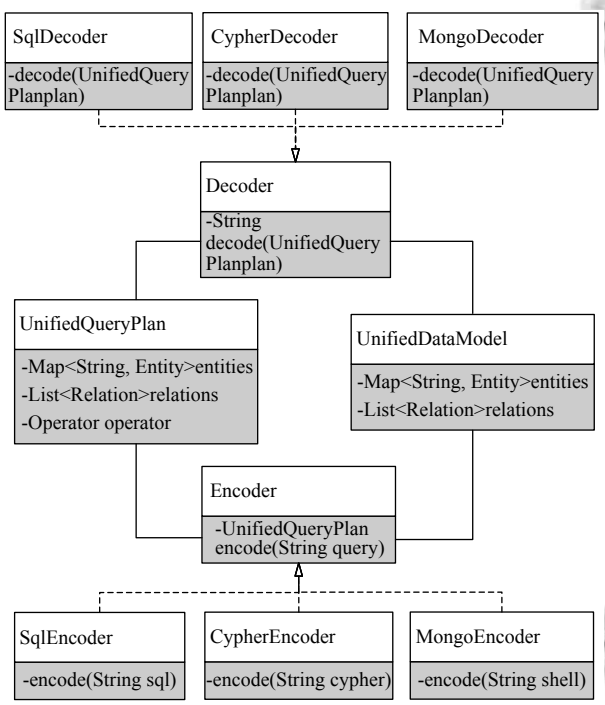

图 3 BAF4DUA 案例工具类

根据 UnifiedQueryPlan 类的 Entities 与 Relations 成员生成 SQL 语句的 SELECT 与 FROM 元素; 通过解析查询计划中的运算 Operator 生成 SQL 语句的 WHERE 条件对结果进行过滤; 类似的, CypherDecoder 主要针对如下 Cypher 句式:

MATCH <ENTITY>
MATCH <RELATION>
WHERE <CONDITION>
RETURN <RESULT>

根据中间查询计划 UnifiedQueryPlan 生成句式的各部分的内容; MongoDecoder 则将 UnifiedQueryPlan 中的 Operator 运算统一组合成一个 Pipeline, 将 Scan、Filter、Link 以及 Project 四类运算转换为 Mongo 命令 Pipeline 中的 JSON 运算对象, 最后将整个 Pipeline 对象序列化为 Mongo Shell 命令语句.

需要指出的是, 本文中案例工具的实现方式存在一定的局限性, 尤其是 Decoder 部分的实现还有很大的提升空间. 例如本文使用了固定的查询语句模式, 根据中间查询计划分别转换为固定句式中的组成部分, 而不是将中间查询计划直接转换为数据提供者查询语句对应的查询计划. 除此之外, 本文的案例没有对聚合查询、多语句查询等操作进行处理.

5.2 查询语言转换性能

为了验证查询语言转换对查询性能的影响, 我们将生成的 1 GB TPCH[16] 数据导入关系型数据库 MySQL、图数据库 Neo4j 以及文档型数据库 MongoDB 中, 对 TPCH 基准测试中单表查询语句 Q1 与多表查询语句 Q2 的查询语言转换与数据查询时间进行了统计, 其中 MySQL、Neo4j 和 MongoDB 数据库的运行环境如表 2 所示, 查询时间结果如表 3 所示. 在数据导入过程中, TPCH 数据集的 8 个文件分别导出为 MongoDB 的 8 个 Collection 以及 Neo4j 的 8 类节点. 根据 TPCH 提供的 ER 模型, 我们在 Neo4j 中创建了节点之间的边. 需要说明的是, TPCH 数据集的不同导入方式以及索引会对后续查询时间产生一定影响.

表 2 数据库运行环境

数据库	运行环境
MySQL	硬件: 8 GB RAM, 200 GB 磁盘, 2核, 虚拟机 软件: MySQL Community 8.0.2
MongoDB	硬件: 8 GB RAM, 200 GB 磁盘, 2核, 虚拟机 软件: MongoDB 4.2.10 Community
Neo4j	硬件: 8 GB RAM, 200 GB 磁盘, 2核, 虚拟机 软件: Neo4j Community 4.1.3

表 3 查询语言转换与数据查询时间 (ms)

查询语句	数据库	查询时间	SQL语言转换时间	Mongo命令转换时间	Cypher语言转换时间
Q1	MySQL	3680	2	3	5
	MongoDB	4856	58	59	61
	Neo4j	4225	3	4	6
Q2	MySQL	967	15	15	20
	MongoDB	6973	64	64	66
	Neo4j	660	17	17	19

© 中国科学院软件研究所 http://www.c-s-a.org.cn

由表 3 可知, 查询语言转换时间在 70 ms 内, 相对于数据查询时间占比最高不大于 1%, 并且随着数据量的增加查询语言转换的时间占比会越来越少. 由于 Mongo 命令的解码过程使用了第三方的 JSON 转换工具, 因此中间查询计划解码为 MongoDB 接口的时间要比其它接口的解码时间更多, 故而查询语言的编码时间均在 10 ms 以下. 总体而言, 在提高开发效率与系统扩展性、满足数据统一访问的前提下, 查询语言转换带来的查询性能损失在可接受范围之内.

5.3 查询语言转换应用案例

案例 1. 数据库缓存: 为提高系统的响应速度, 应用系统通常采用主数据库与缓存数据库相结合的方式管理业务数据, 例如使用 Redis 对用户信息进行缓存. 此时系统开发需同时使用 SQL 与 Redis 语言进行数据查询, 如果能够将 SQL、Mongo 命令以及 Cypher 编码为中间查询计划, 同时将中间查询计划解码为 Redis 查询语句, 那么系统就可以为不同主数据库增加缓存功能, 并且保持系统开发的接口统一性.

为实现以上目标, 系统需要一个数据提供者端 Adapter 将中间查询计划转换为 Redis 查询语言. 其中数据映射将 Redis 数据库中具有一定模式的 Key 定义为中间数据模型的 Entity; Entity 的属性值与 Key 值相对应; Relation 则根据用户定义表达式进行 Entity 的关联. 中间查询计划与 Redis 的具体运算映射方式如表 4 所示.

表 4 中间查询计划与 Redis 运算映射

查询计划	Redis运算
Scan	将Entity对应的Key模式作为查询Key的条件
Filter	将Filter的表达式转换为Key的模式, 添加到Key的查询条件中
Link	自定义补偿运算实现两个Entity关联
Project	自定义补偿运算实现Entity、Relation以及Attribute的投影

以客户 Customer 数据为例, 假设客户数据在 Redis 中 Key 的存储模式为<*Cust_ID_USERNAME_NATION, customer_object_hash*>, 那么 Customer 实体定义为以字符串“Cust”开头的 Key, 实体属性 ID、USERNAME、NATION 的值分别对应以“_”进行分割的每个值. 如图 4 所示, 假设系统要查询“李”姓中国籍客户, 那么使用 SQL、Cypher 以及 MongoDB 命令进行查询时会首先编码为中间查询计划, 然后再转换为 Redis查询语句, 最后实现 Redis 对 3 类主数据库的缓存.

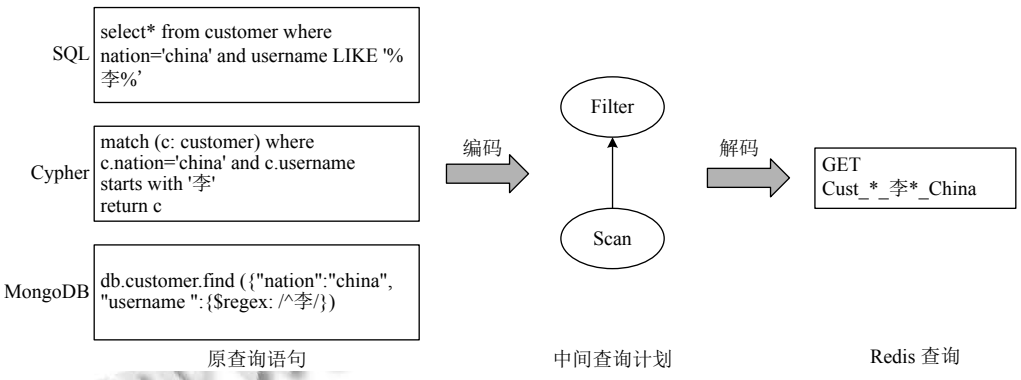

图 4 Redis 查询语言转换案例

案例 2. 数据统一管理: 科学数据管理系统通常会查询多个数据源, 甚至会调用不同学科的数据源来验证假设的真伪. 国家重点研发计划“科学大数据管理系统”项目为用户提供包括天文、高能物理和微生物 3 种异构数据在内的多元异构数据的统一访问接口. 以高能物理数据处理系统 EventDB[17] 为例, 它通过分布式存储与数据索引, 能够对高能物理实验中产生的 EB 级事例进行存储与查询. EventDB 的接口形式为特定命令, 如有以下查询命令: *time*=178789800~178807800&*detID*=1&*channel*=12&*pulse*=0~255&*eventType*=0, 其含义为查询 2017 年 8 月 31 日 15:50:00 至 2017 年 8 月 31 日 20:50:00 之间 5 小时内 1 号探测器、12 号通道、0 号事件的脉冲跨度的变化情况.

数据提供者 Adapter 的数据映射需根据科研用户的查询需求和习惯, 根据实验因素与中间模型进行映射. 例如相同特征的事例映射与 Entity 集合对应; 实体属性为 EventDB 中事例的相关数据值; Relation 根据自定义表达式进行 Entity 关联. 中间查询计划与 EventDB

© 中国科学院软件研究所 http://www.c-s-a.org.cn

命令的映射如表 5 所示.

假设用户习惯按照探测器进行事例查询, EventDB 中的探测器编号为 *n* 的事例与 Entity 映射, 事例的 time、detID、channel、pulse 与 eventType 等属性为 Entity 的属性值. 如图 5 所示, 如果要查询 3 号探测器在 2017 年 8 月 31 日 15:50:00 至 2017 年 8 月 31 日 20:50:00 之间内的所有脉冲变化, 那么 SQL、Cypher、MongoDB 的查询语句可根据以上映射关系首先编码为中间查询计划, 然后转换为 EventDB 查询命令.

表 5 中间查询计划与 EventDB 运算映射

查询计划	EventDB运算
Scan	将Entity对应的事例特征作为查询条件
Filter	将Filter的表达式添加到查询命令条件中
Link	自定义补偿运算实现两个Entity连接
project	自定义补偿运算实现Entity、Relation以及Attribute的投影

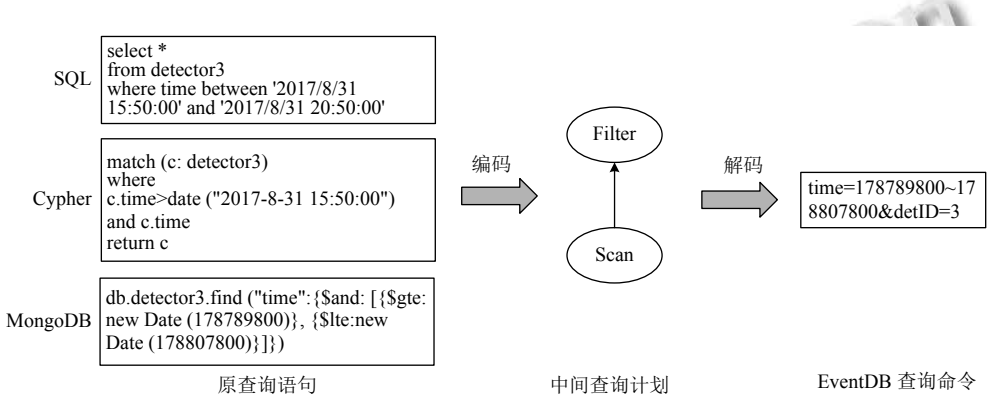

图 5 EventDB 查询语言转换案例

6 结论与展望

在大数据时代, 集成系统使用多样化工具对内外部数据进行统一管理分析, 这增加了系统开发、数据管理工具更新迁移以及数据统一管理等任务的接口适配成本. 本文提出了一种基于查询语言转换的多源数据统一访问框架 BAF4DUA, 该框架特色之处在于采用了双 Adapter 模式: 数据消费者 Adapter 先将消费者数据访问接口语言转换为中间数据模型的查询计划, 而后通过数据提供者端 Adapter 将中间查询计划转换为数据提供者的数据访问接口. 通过 BAF4DUA 框架, 可实现数据消费者的查询语言接口与数据源的数据模型相分离, 数据消费者和数据提供者可采用即插即用的方式增加或更新相应的 Adapter, 增加了系统的灵活性与扩展性, 更符合集成系统多对多数据访问的特点.

本文所提 BAF4DUA 框架的局限性在于仅考虑了单一使用场景下的数据查询操作, 还需要进一步完善以适应更复杂的数据统一访问应用场景. 后续研究可从以下几个方面进一步拓展: (1) 设计与实现支持具备 count、sum 等更丰富功能的查询运算类型; (2) 设计与实现查询数据位于不同数据库场景下的数据处理方案; (3) 设计与实现智能化数据统一插入存储方案.

参考文献

1 Lenzerini M. Data integration: A theoretical perspective. Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Madison, WI, USA. 2002. 233–246.

2 Stonebraker M, Cetintemel U. “One size fits all”: An idea whose time has come and gone. Proceedings of the 21st International Conference on Data Engineering. Tokyo, Japan. 2005. 2–11.

3 黎建辉, 沈志宏, 孟小峰. 科学大数据管理: 概念、技术与系统. 计算机研究与发展, 2017, 54(2): 235–247. [doi: 10.7544/issn1000-1239.2017.20160847]

4 Rachapalli J, Khadilkar V, Kantarcioglu M, *et al*. RETRO: A framework for semantics preserving SQL-to-SPARQL translation. Proceedings of the 15th Annual International Symposium on Wearable Computers. San Francisco, CA, USA. 2011. 1–16.

5 Eisenberg V, Kanza Y. D2RQ/Update: Updating relational data via virtual RDF. Proceedings of the 21st International Conference on World Wide Web. Lyon, France. 2012. 497–498.

6 Steer BA, Alnaimi A, Lotz MABFG, *et al*. Cytosm: Declarative property graph queries without data migration. Proceedings of the 5th International Workshop on Graph

© 中国科学院软件研究所 http://www.c-s-a.org.cn

Data-management Experiences & Systems. Chicago, IL, USA. 2017. 1–6.

7 Armbrust M, Xin RS, Lian C, *et al*. Spark SQL: Relational data processing in spark. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne, VIC, Australia. 2015. 1383–1394.

8 Sethi R, Traverso M, Sundstrom D, *et al*. Presto: SQL on everything. Proceedings of 2019 IEEE 35th International Conference on Data Engineering. Macao, China. 2019. 1802–1813.

9 Kornacker M, Behm A, Bittorf V, *et al*. Impala: A modern, open-source SQL engine for hadoop. Proceedings of the 7th Biennial Conference on Innovative Data Systems Research. Asilomar, CA, USA. 2015. 1–10.

10 Junghanns M, Petermann A, Gómez K, *et al*. GRADOOP: Scalable graph data management and analytics with Hadoop. arXiv: 1506.00548, 2015.

11 O'Brien K, Gadepally V, Duggan J, *et al*. BigDAWG polystore release and demonstration. arXiv: 1701.05799, 2017.

12 Chen PPS. The entity-relationship model—Toward a unified view of data. ACM Transactions on Database Systems, 1976, 1(1): 9–36. [doi: 10.1145/320434.320440]

13 Calcite. https://calcite.apache.org/. [2020-12-25].

14 Cytosm. https://github.com/cytosm/cytosm. [2020-12-25].

15 Mongo-Shell-Like-Query. https://github.com/EqualExperts/mongo-shell-like-query. [2020-12-25].

16 TPC. TPC-H is a Decision Support Benchmark. http://www.tpc.org/tpch. [2020-12-25].

17 程耀东, 张潇, 王培建, 等. 高能物理大数据挑战与海量事例特征索引技术研究. 计算机研究与发展, 2017, 54(2): 258–266. [doi: 10.7544/issn1000-1239.2017.20160939]

© 中国科学院软件研究所 http://www.c-s-a.org.cn