第33卷 第17期 计 算 机 工 程 2007年9月
Vol.33 No.17 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)17—0111—03 文献标识码：A 中图分类号：TP311.13

基于异构集成的元数据及其多表动态查询算法

李润洲，方 明
(西安石油大学计算机学院，西安 710065)

摘 要：基于企业内多个分布异构关系数据库的集成需求，设计了一个向上支持集成访问界面，向下表述数据库网络位置、数据模式、数据内容的元数据字典模式。给出了面向集成环境和各异构数据库的通用查询请求表示。基于元数据字典，提出了一种因访问需求变化而引起相关数据库关系表动态变化的动态查询语句构造算法，并对算法进行了论证。

关键词：关系数据库；元数据；数据集成；动态查询

Metadata and Dynamic Multi Tables Query Algorithm Based on Heterogeneous Integration

LI Run-zhou, FANG Ming
(School of Computer Science, Xi'an Shiyou University, Xi'an 710065)

【Abstract】Based on the integration demand of multi distributed and heterogeneous relational databases within an enterprise, a metadata dictionary pattern is designed to sustain upper integration access interface and to depict the lower databases' network location, data schema and data content. A denotation of universal query request orienting integration environment and heterogeneous database is given. Based on the metadata dictionary, a dynamic query algorithm to join relevant database's multi tables changing with the change of access need involved in a query is put forward, and the algorithm is demonstrated.

【Key words】relational database; metadata; data integration; dynamic query

企业在长期生产实践中积累起来的信息数据虽然大多以关系数据库的形式存储，但企业内各部门相对独立的管理职能往往使得各部门能够自主研发数据库、选择数据库产品，并部署在符合各自生产和应用需求的计算平台上。如何消除各个数据库在数据模式、访问方式等方面的差异，为企业内各类用户提供集成的信息服务，提高资源的利用率，是当前企业信息化迫切需要解决的问题之一。在集成多个分布、异构关系数据库的应用环境中，信息数据查询机制需要解决的问题包括：(1)为用户提供消除了属性模式冲突的集成访问界面，允许用户提出基于集成访问界面的查询请求；(2)获取关系数据库的位置、模式、数据内容以及关系间的关联等非数据信息的机制；(3)基于获取的非数据信息，分解面向集成环境的查询，动态生成面向具体数据库的查询语句。

针对上述问题，本文建立了一个表示分布异构环境中各数据库结构信息、内容信息的元数据字典，然后基于元数据字典提出一个转换通用查询命令为符合 ANSI 规范的 Select 语句的动态构造算法，并对随查询应用需求不同而引起的相关关系表连接表达式的动态生成方法进行了讨论。其中，元数据字典及通用查询命令采用 XML 文档存储表示，动态查询语句生成算法在.NET 环境中实现。

1 元数据字典的组成与结构

元数据是描述数据的数据[1]，本文将关于数据库位置、模式、内容等的描述信息称为元数据。为获取元数据，可直接访问每个涉及的DBMS维护的系统表[2]。但是在异构环境中，系统表的访问方法随DBMS的不同而不同，并且提供的信息模式也会存在很大差异、信息内容零散且难以组织，这为动态构造查询语句造成了困扰，不利于通用性编程。而针对特定数据库查询语句的构造方法，文献[3,4]提出了一种查询驱动的灵活数据库设计，组合所有可能的查询连接，但是它要维护大量并行的只读数据库结构。

针对用户基于数据位置、数据命名和数据模式等方面的透明性集成访问需求，集成系统数据访问界面应该能够给用户提供消除了模式中属性描述冲突的抽象公共可查询逻辑实体。当用户基于该逻辑实体属性提出查询请求时，集成系统能将这一面向集成环境的查询分解、映射到具体的数据库，再针对特定数据库，生成符合该数据库的查询语句。这一过程依赖于集成系统中各数据库相关信息的良好表达，为此，本文通过构建分布异构数据集成应用框架下的元数据字典，来表示、存储、访问这些非数据信息，并提供元数据访问引擎，为应用检索元数据、分解查询、构造查询语句提供必要的支持。

抛开各数据库外部技术表现的不同以及数据结构的差异，集成环境中的各数据库所记录的业务内容存在着相关性。为了有序组织分散的数据内容，元数据字典基于数据内容，从多个数据库中抽象出多个可查询的业务实体，定义它可能的，以及应该拥有的属性，用形象且直观的命名机制予以命名，作为面向用户的可查询逻辑实体。例如，某企业的多个管理部门中都可能存储了员工的人事资料，使用不同的管理

作者简介：李润洲(1972－)，女，讲师、硕士，主研方向：计算机网络，管理信息系统；方 明，教授、博士后
收稿日期：2006-12-15 **E-mail：**Runrunli@126.com

系统、不同的数据库结构，提供不完全相同的信息内容，本文从几个数据库中抽象出一个公共的逻辑业务实体“员工人事信息查询”,定义它所有可能涉及的属性以及到各个数据库的映射关系，所有涉及这类业务实体的查询，都将引用该逻辑实体，而不必分别引用不同的数据集合。逻辑实体元数据结构如图 1 所示，其中，viewColumns 元素指出逻辑实体可视的逻辑属性；mapSchema 元素用来指出逻辑属性与物理属性的映射关系，是一个较复杂的元素，以存储设备、数据库、关系表、物理字段的路径来确定一个逻辑属性所映射到的物理属性，包括物理属性所在的存储设备、数据库、表、物理字段，为分解、映射面向集成环境的查询过程提供支持。

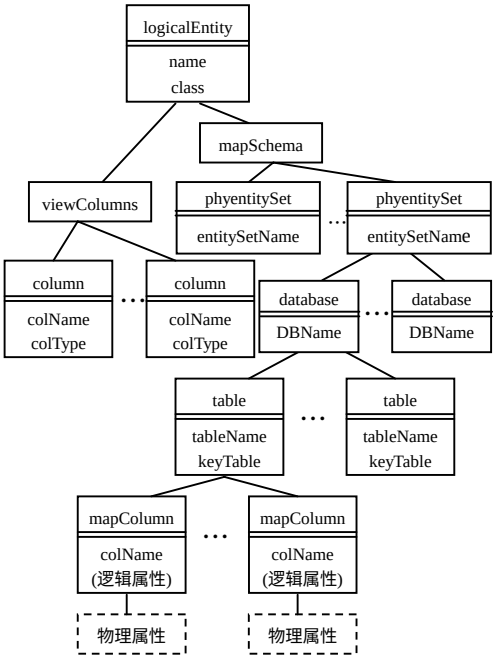

图 1　逻辑实体元数据结构

另一方面，元数据字典也详细记录着每个物理数据库的相关信息，包括网络位置、访问接口、数据模式等信息，具体到关系表、字段。每一个物理数据库构成一个物理实体，由物理实体元数据描述，结构如图 2 所示。

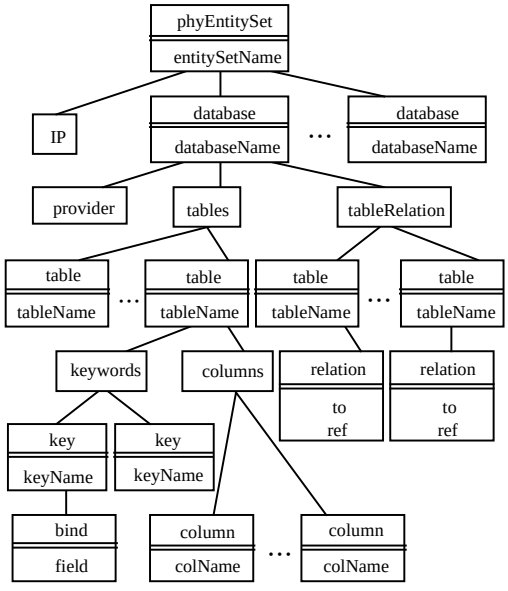

图 2　物理实体元数据结构

图 2 中除了表示数据库网络位置的 IP 元素、数据库编程访问接口的 provider 元素以及描述数据库内关系表的 table 元素之外，还使用 relation 元素来描述各个表之间的关联关系。对每一个关系表，relation 元素的 to 属性指出表的外键，ref 属性指出外键关联的表名，可通过表名来找到表所定义的主键而建立引用关系，为应用程序实现特定数据库查询语句的构造过程提供帮助。

物理实体是和底层数据库数据源紧密关联的数据对象，但是通过逻辑实体的 mapSchema 元素所建立的逻辑实体属性到物理属性的映射关系，向高层屏蔽了底层数据的组织结构、命名机制等特性，并且保证在更新数据库模式时，只需要更新这一映射模式，这确保了物理实体的相对独立性，能满足将来技术发展、数据库内容更新及不断发展的需要。

2　查询请求和返回结果表示

用户基于访问界面提出的查询请求，面向逻辑实体，应用程序首先需要依据逻辑实体的映射模式将它映射为面向物理实体的查询请求，之后再将它转换为符合 ANSI 规范的查询语句。为了在异构平台间传递信息，查询过程中各个阶段的查询请求以及返回的查询结果集都通过生成 XML 文档交互。XML 查询请求文档包含 2 部分内容：查询命令和查询的信息实体。DBSelect 构成请求文档的根元素，它的信息实体是操作的对象，可能面向逻辑实体，也可能面向物理实体，在访问深入的过程中，由应用程序逐步更新。各个阶段生成的请求文档，包括面向逻辑实体的查询请求、面向多个物理实体的查询请求以及分解形成的单独数据库查询请求，都使用相同的 DTD(document type definition)定义，如下所示。

```
<!ELEMENT DBSelect (Xobject+)>
<!ELEMENT Xobject (from+)>
<!ATTLIST Xobject
Xname ID #REQUIRED>
<!ELEMENT from (select?,where?)>
<!ATTLIST from
        name CDATA #REQUIRED
        keyTable (true|false) #IMPLIED>
<!ELEMENT select (column+)>
<!ELEMENT where (object)>
<!ELEMENT column (#PCDATA)>
<!ELEMENT object (#PCDATA)>
<!ATTLIST object
        colName CDATA #REQUIRED
        comOperate CDATA #REQUIRED
operName (NULL|NOT) #IMPLIED>
```

面向逻辑实体的查询请求文档中，Xobject 元素和它的子元素 from 都指出逻辑实体名，而映射为物理实体的查询请求文档或分解到具体数据库的查询请求文档中，Xobject 元素指出数据库名，from 元素指出表名，它的 select 和 where 子元素则分别指出查询的内容和查询条件。

返回文档的根元素为 DBReply，它的信息实体面向查询涉及物理实体的物理属性集。由于各数据库在属性模式描述之间的冲突，目前的实现仅将各个数据库基于单个条件的查询结果集采用运算后返回给用户。

3　基于元数据的查询语句构造算法

3.1　算法思想

定义 1　一个数据库关联图G=<V,E>是一个有向图，其中，$V=\{V_1,V_2,\ldots,V_n\}$，表示数据库的数据表，　$E=\{<V_i, V_j>| <V_i, V_j>$表示当表$V_j$的外键引用表$V_i$的主键时，$V_i$到$V_j$存在一条有

向边}，即E是数据表关联集合。一次查询涉及数据表的关联图 $G'=<V',E'>$，是图G的子图。

定义 2 设数据库关联图为 G=<V,E>，n=|V|，并假定各结点是有序的，使

$$m_{ij}=\begin{cases}1, & \text{如果} <V_i, V_j> \in E\\ 0, & \text{如果} <V_i, V_j> \notin E\end{cases} \quad (i, j=1,\ldots,n)$$

则称$\boldsymbol{M}[m_{ij}]$矩阵是关联图G的关联矩阵。

定义 3 在数据库关联图G=<V, E>中，如果$<V_i, V_j> \in E$，则称V_i为V_j的引入表。设一次查询涉及数据表的关联图为$G'=<V',E'>$，对任一个$V_i \in V'$，使$Q_j=\sum m_{ij}$，那么，Q_j表示表V_j在待查询表集内的引入表个数，称Q_j为V_j的净入度。

用户提出的查询请求被分解到特定数据库后，所涉及关系表随应用需求的不同而呈现出动态变化的特性，算法必须依据物理实体元数据所记录的数据表之间的关联关系，动态构造连接表达式，连接多个表，生成多表连接的查询语句。但多表连接时查询主体的变化将导致返回结果集的极大差异。为此，对每个可查询逻辑业务实体，要求确定一个查询主体，或者由用户指定，然后映射到一个具体的表，或者在元数据字典中设定一个默认主表(见图 1 table 元素的 keyTable 属性)。

算法从主表出发，作前向探索[5]，即在待查询表集内查找它的引入表，选择一个引入表作工作表，继续查找，直到某一时刻，工作表的净入度为 0。然后作回退处理，将零净入度表与前一个工作表合并，对前一工作表的所有引入表作相同处理，直到将工作表与它的所有引入表合并为一个结点，返回上一级表。对上一级表作相同处理，直到主表。

3.2 算法设计

(1)生成数据库的关联矩阵。

(2)创建集合 S=Φ, F=Φ, W={条件表达式}。

(3)依据面向物理实体的查询请求文档，生成待查询表集合 Q，确定查询主表，将查询主表作为当前工作表 T。

(4)T 压栈，Q=Q-{T}。

(5)在 Q 中查找 T 的引入表 I，如果 I 存在，则把 I 作为当前工作表 T，转(4)；否则，弹栈，将弹出表作为零净入度表 Z，如果栈空，则转(7)；否则转(6)。

(6)将 Z 的查询字段与栈顶表合并，在查询条件 W 集合中并入栈定表到 Z 的等值连接，删除有向图 G' 中栈顶表到 Z 的有向边，将处理后的栈定表作为当前工作表 T，Q=Q⋃{Z}，转(5)。

(7)合并 S, F, W 集合，生成 Select 语句结束。

在整个算法演算过程中，创建了 3 个集合 S, F, W，用于构造 Select 查询语句的 Select, From 和 Where 子句。S 和 F 集合初值为空，W 集合的初值为映射后面向物理实体的查询条件。算法步骤(7)中，零净入度表与栈顶表的合并过程工作包括：1)给 S 集合并入零净入度表经表名限定修饰后的查询字段。2)栈顶表的被检索外键替换为零净入度表的主键，将它并入 S 集合。如果栈顶表的引入表是代码表，如员工人事信息查询中，员工基本信息表引入的员工部门表的主键是部门代码，为了能在结果集中反映实际的部门名称，在元数据字典中，对这一类引入设定绑定 bind 字段(见图 2 的 key 元素)，并入到 S 集合中的引入表字段就是绑定字段而非主键。3)将零净入度表的表名并入 F 集合。W 集合中初始为映射后面向物理实体的查询条件，以后逐步将栈顶表外键与引入表主键的等值连接表达式并入 W 集合。生成查询语句时，将 W 集合中的各分量使用 AND 连接构成查询语句的 Where 子句。

3.3 算法分析

如果一次查询分解到某个数据库时涉及关系表的关联图构成了一个回路，如图 3 所示，那么同时连接查询表 1 和表 2，可能大量削减返回记录数量而破坏原始查询目的，因此，在算法中，对这一类循环引用实施了断开回路处理，同时也确保了算法能够正常结束。

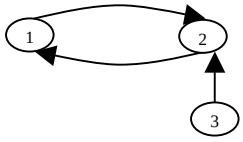

图 3 构成回路的表间关联图

算法中，引入表的查找范围为集合 Q，每一个工作表 T 压栈时，都从 Q 中减去 T，这保证了在前向探索过程中不会形成循环引用。

证明 假设在前向探索的过程中形成了循环引用，则压入堆栈的表结点间至少形成了一个回路，也即堆栈中至少有 2 个相同的表，设这 2 个表为 T，那么这与 Q=Q-{T}在 Q 中查找 T 的引入表矛盾。因此，在前向探索的过程中，不会形成循环引用。

将工作表 T 与它的一个引入表 Z 合并处理后，删除了 T 到 Z 的有向边，这使得任一条路径的探索过程都不会再次重复使用这个连接关系，确保了在回退处理时也不会形成循环引用，保证算法能够正常结束。之后，将 Z 并入 Q 集合，使得其他表能够从其他路径正确连接到表 Z，完成 T 的所有引入表连接。

在测试系统中，生成的 Select 语句基于 ANSI 标准，依据元数据字典提供的数据库位置和驱动程序信息，使用 ADO 动态连接关系数据库，完成查询访问。限于篇幅，具体的实现代码略去。

4 小结

针对分布异构环境中位置透明、命名透明、模式透明的集成访问需求，本文提出了抽象各数据库相关信息和信息内容的元数据字典，设计了基于元数据字典的动态查询语句构造算法，讨论动态多表连接查询语句生成方法，这一算法在笔者的实验系统中已获得验证。在下一步的工作中，将就如何解决异构环境中属性模式冲突，在不同的数据库属性之间实现算术运算、函数运算、分组、排序等复杂查询条件，进一步展开研究。

参考文献

1 Marco D. 元数据仓储的构建与管理[M]. 张 铭, 李 钦, 译. 北京: 机械工业出版社, 2004.

2 何珍文, 吴冲龙, 张夏林. 等. 数据库应用程序中通用动态查询实现方法研究[J]. 计算机工程, 2002, 28(11): 92-94.

3 Chen A, Goes P, Gupta A, et al. Heuristics for Selecting Robust Database Structures with Dynamic Query Patterns[J]. European Journal of Operational Research, 2006, 168(1): 200-220.

4 Chen A, Goes P, Marsden J. A Query-driven Approach to the Design and Management of Flexible Database System[J]. Journal of Management Information System, 2002, 19(3): 121-154.

5 卢开澄, 卢华明. 图论及其应用[M]. 2 版. 北京: 清华大学出版社, 1995.