第35卷第5期　　中国科学技术大学学报　　Vol. 35, No. 5
2005年10月　　JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA　　Oct. 2005

文章编号:0253-2778(2005)05-0674-09

利用关系数据库系统对半结构化数据进行近似查询*

韩　恺,岳丽华,龚育昌

(中国科学技术大学计算机科学与技术系,安徽合肥 230026)

摘要:提出一种利用关系数据库系统在一般图结构的半结构化数据上进行近似查询的途径.根据嵌套结构和文本值的相似性来度量路径的相似性;根据路径的相似性得到查询目标节点与数据源节点的相似性.为返回数据源中与查询目标节点相似的节点,首先提取出数据源中长度在固定范围内的所有路径,然后利用关系数据库系统将其与查询路径进行相似性连接,并按相似度从大到小返回所有结果.为提高相似性连接的效率,引入q窗口概念,并利用若干路径相似的必要条件来减少计算相似性函数的次数.试验证明了其有效性.

关键词:半结构化数据;图结构;近似查询;相似性度量;相似性连接;关系数据库系统

中图分类号:TP311　　**文献标识码**:A

0　引言

半结构化数据的查询技术是当前研究的热点之一.半结构化数据具有异构性和无模式性,要求返回精确的查询结果往往是不够的.实际上,由于半结构化数据缺少模式信息,以及经常存在动态改变的情况,用户往往不能准确把握数据源的结构,并且一个查询与数据源中满足条件的查询结果之间存在结构以及文本值上的差异是很常见的.因而,我们希望引入信息检索技术的思想,使得能够在半结构化数据源中检索出可能的查询结果,并将查询结果根据与原查询的相似性程度排序来返回给用户.本文就是针对这一问题展开研究的.

我们利用最长的顺序标签子序列来度量查询路径与数据源路径的嵌套结构的相似性,并利用Levenstein距离[1,2]度量路径尾节点所带文本值内容的相似性,根据路径的相似性得到节点的相似性度量.为返回数据源中所有与查询目标节点相似的节点,我们根据相似性度量得到可能与查询路径相似的路径长度范围,然后在数据源中提取出在该范围内并且尾节点带有文本值的所有路径.利用关系数据库将其与所有的查询路径进行相似性连接从而得到所有的相似路径对.为了提高相似性连接的效率,引入q窗口概念,并根据若干过滤条件来减少计算代价较高的相似性函数的次数.最后,利用一条关系查询语句得到所有与查询

* **收稿日期**:2003-10-28;**修回日期**:2004-04-28
基金项目:中科院知识创新项目(K2CX0101).
作者简介:韩恺,男,1977年生,博士生. E-mail:kaihan@ustc.edu

相似的查询结果并按相似度从大到小的顺序返回给用户.

目前,对于半结构化数据的查询研究多集中于为获取精确查询结果的查询技术[3~7],对半结构化数据近似查询的研究较少[8,9]. Schlieder[8]提出一种带有代价的改进树嵌入算法来实现对 XML 文档的近似查询. Amer-Yahia 等人[9]将可能的结构变化编码在查询树中,在查询执行过程中根据阈值及时地剔除结构变化过大的可能查询结果,以提高近似查询的效率. 文献[8]和[9]提出的近似查询方法均建立在源 XML 文档为树的前提下,实现基础是对 XML 树的结构连接操作[4,5],而对于一般非树结构的半结构化数据,由于结构连接并不能实现,因此他们的方法并不适用. 此外,该方法仅考虑了查询结构与数据源的相似性,未考虑查询文本值内容与数据源的相似性. 而本文提出的方法能够对一般的有向图结构的半结构化数据进行近似查询,并同时考虑了查询嵌套结构及其文本值内容与数据源的相似性,提出了利用关系数据库来对一般图结构的半结构化数据进行近似查询的工作.

1　基本定义

在本节中我们给出关于数据源、查询和路径的基本定义.

定义 1　半结构化数据是一个带有根节点有向图 $G=<V, E, \mathrm{Oid}, \mathrm{Label}>$,其中 V 是节点的集合,E 是边的集合,$V=V_c \cup V_x$,集合 V_x 中的任意节点 u 称为一个值节点,其带有一个文本值,记为 $\mathrm{Value}(u)$. 而 V_c 中的节点不带有文本值,称为普通节点. 对于任意节点 $v \in V$,$\mathrm{Oid}(v)$是节点 v 的惟一标识符,对于任意 $e=<v_1, v_2> \in E$,$\mathrm{Label}(e)$是节点 v 的标签. v_1称为 e 的头节点,v_2 称为 e 的尾节点. 记 $\mathrm{root}(G)$为 G 的根. G 中存在惟一一条头节点为空,尾节点为 $\mathrm{root}(G)$的边,该边称为 G 的根边. 图 1(a)显示了一个半结构化数据源的有向图模型.

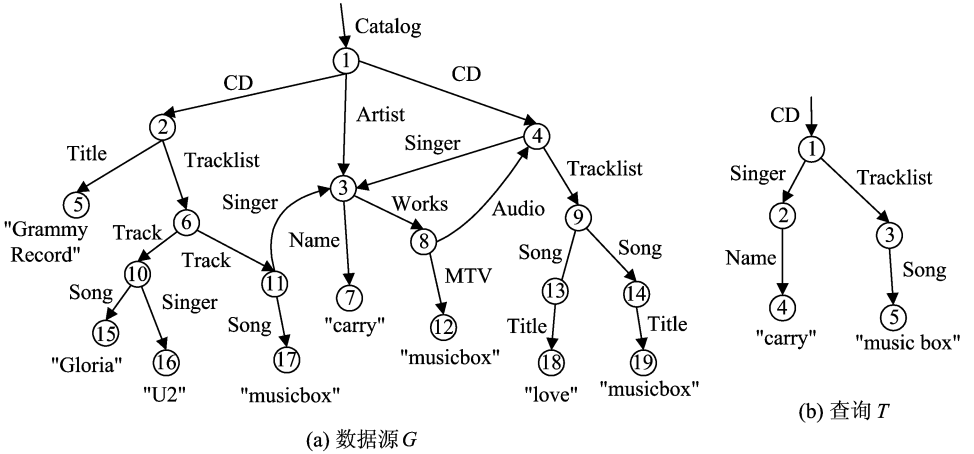

图 1　数据源 G 和查询 T

Fig. 1　Data source G and query T

半结构化数据的查询的定义与定义 1 类似,所不同的是每个查询都是一棵树. 图 1(b)显示了一个查询,该查询的含义是"返回所有包含歌手 carry 所唱的歌 music box 的 CD". 我们把查询树的根节点称为该查询的目标节点.

定义2 有向图G中的一条路径p定义为$e_1.v_1.e_2.v_2\cdots.e_n.v_n$,其中边$e_1$的尾节点为$v_1$,称为$p$的起始边. label($e_1$)称为该路径的起始标签. 边$e_i(2\leqslant i\leqslant n)$是$G$中从$v_{i-1}$到$v_i$的有向边. v_1称为p的起始节点,记为Start(p),v_n称为p的尾节点,记为End(p). 字符串序列$<$Label(e_1),Label(e_2),$\cdots$Label(e_n)$>$称为p的标签序列,记为LS(p). 如果End(p)是一个值节点,则路径p称为一条完整路径. 如果完整路径p的起始边为G的根边,则称p为一条全路径.

2 相似性度量

为在半结构化数据源G中根据查询T进行近似查询,必须找到一种相似性度量方法来衡量G中的每个节点与查询T的目标节点的相似. 下面引入相似性度量方法.

2.1 路径相似性

导致一个查询与数据源不能完全匹配的主要原因往往是用户提交的查询不能与数据源保持嵌套结构的完全一致. 这种情况造成了查询中的路径与数据源中的路径在标签序列上具有"大致相同"的特点. 根据公共的顺序子标签序列来考察查询中的路径与数据源中的路径的相似性. 首先给出字符串序列的顺序子序列的定义:

定义3 字符串序列$A=<A_1,A_2,\cdots A_n>$的保持元素先后顺序的子序列$<A_{m_1},A_{m_2},\cdots A_{m_k}>(m_1<m_2<\cdots<m_k,1\leqslant m_i\leqslant n,1\leqslant i\leqslant k)$称为$A$的一个顺序子序列. 对于两个字符串序列$A$和$B$,其最长的相同顺序子序列的长度记为MaxSub($A$,$B$).

假设A和B的长度分别为n和m,则计算MaxSub(A,B)可以利用如下的动态编程算法[10]:维持一个$(n+1)\times(m+1)$维矩阵L,并令$L[i,j]=$MaxSub($<A_1.A_2\cdots A_i>$,$<B_1.B_2\cdots B_j>$),$L[i,0]=L[0,j]=0$,其中$1\leqslant i\leqslant n$, $1\leqslant j\leqslant m$,则$L[n,m]$即为所求结果. L的计算可以通过如下二重循环得到,该算法的复杂度为$O(mn)$.

如果$A_i=B_i$,则$L[i,j]=L[i-1,j-1]+1$; $L[i,j]=\max\{L[i,j-1],L[i-1,j]\}$; $(1\leqslant i\leqslant n,1\leqslant j\leqslant m)$.

根据定义3,假设有两条路径$p1=e_1.v_1.e_2.v_2\cdots.e_n.v_n$,$p2=u_1.o_1.u_2.o_2\cdots.u_m.o_m$,令$l=$MaxSub(LS($p1$),LS($p2$)),$t=2l/(m+n)$,显然,$t$的值越大,说明这两条路径在嵌套结构上越相似. 而$t$值若太小,则说明这两条路径代表不同的语义. 因此,我们可以用t来作为衡量路径相似性的一个参数.

考察两条完整路径的相似性时,还必须考虑路径尾节点的文本值的相似性. 例如,考察图1中的查询路径CD.1.Tracklist.3.Song.5,其尾节点的值为"music box". 因此,即使在数据源中有一条标签序列与该路径相同的完整路径(此时$t=1$),假如其尾节点的值与路径p的完全不同,例如为"Gloria",显然这两条路径表示不同语义,不应为相似的路径. 另一方面,考察尾节点文本值的相似性必须能够容忍用户所给的文本值具有轻微的错误的情况,例如上述查询路径尾节点的值为"music box",然而在数据源中该值应该为"musicbox". 综上所述,判断文本值的相似性类似于数据清洗研究领域中的"字段匹配"问题[1,11]. 我们采取流行的Levenstein距离[1,2]来衡量两个文本值的相似性. 假设两个文本值的长度分别为k和l,则计算Levenstein距离的时间复杂度已知为$O(kl)$.

最后,由于一条路径的起始标签规定了该路径所描述的实体对象,因此,如果两条路径

的起始标签不相同,则它们应看作不相似的.下面给出判断两条完整路径是否相似的算法:

算法 1　判断完整路径相似性

输入——两条完整路径 $p1$,$p2$,长度分别为 m 和 n,阈值 δ 为非负整数,阈值 θ 为[0,1]之间的实数;

输出——若两条路径相似,则返回 true,否则返回 false;

```
Boolean PathSim(p1, p2, δ, θ)
  begin
    if (p1 与 p2 的起始标签不同)  then  return false;
    计算 Value(End(p1))和 Value(End(p2))的 Levenstein 距离 s;
    if s>δ then return false;
    计算 t= 2MaxSub(LS(p1),LS(p2))/(m+n);
    if  t <θ  then  return false;
    return true;
  end
```

根据算法 1,设完整路径 $p1$ 和 $p2$ 的长度分别为 n 和 m,Value(End($p1$))和 Value(End($p2$))的长度分别为 k 和 l,则算法 1 的复杂度为 $O(\max(mn,kl))$.

2.2　节点相似性

根据以上路径相似性可以给出查询 T 的目标节点 root(T)与数据源 G 中的节点 v 的相似性.直观的说,假如 T 中有足够多的全路径与 G 中以 v 为起始节点的完整路径相似,则我们可以定义 v 与 root(T)的相似度为这些相似路径的个数.定义 4 说明了这一点.

定义 4　对于一个查询 T,令集合 Z 为 T 中所有全路径的集合.对于数据源 G 中的节点 v,令 $J=\max\{|X|\ |X\subseteq Z$,对于任意 $y\in X$,存在 G 中的完整路径 g 满足 Start(g)$=v$,并且路径 y 与 g 相似 $\}$,则节点 v 与 root(T)的相似度可以定义为

$$\mathrm{Sin}(v,T)=\begin{cases}0 & \text{当 } J<\lambda|Z|;(\text{阈值 }\lambda\text{ 是}(0,1]\text{ 之间的实数})\\ J & \text{otherwise};\end{cases}$$

3　近似查询

有了以上的相似性度量方法,现在我们可以确切定义半结构化数据的相似性查询问题.即:给定数据源 G 和查询树 T,找到 G 中与 root(T)的相似的所有节点,并按相似度由大到小的顺序排列将其返回给用户.

为解决以上近似查询问题,假设查询 T 中所有全路径的集合为 Z,步骤为:(1)路径提取:抽取出 G 中所有可能与 Z 中的路径相似的路径集合 P.(2)相似性连接:根据算法 1 的路径相似度量方法找出 P 与 Z 中所有相似的路径对.(3)得到查询结果:根据得到的相似路径对找到 G 中所有与 root(T)相似的节点,并根据相似度从大到小的顺序返回给用户.

3.1　路径提取

为进行相似查询,首先我们必须提取有向图 G 中所有可能与查询 T 中的全路径相似的完整路径.下面引入定义 5 和引理 1.

定义 5　对于任意两个字符串序列 $A=<A_1,A_2,\cdots A_n>$,$B=<B_1,B_2\cdots B_m>$,总可以

通过若干添加/删除元素的操作来将 A 转化为 B(例如可以先删除 A 中所有元素,再按序添加 B 中所有元素). 我们把将 A 转化为 B 所需的最少的添加/删除操作次数称为字符串序列 A 和 B 的转换距离,记为 $D(A,B)$(显然有 $D(A,B)=D(B,A)$). 并把任意将 A 转换为 B 的 $D(A,B)$个插入/删除操作序列称为 A 到 B 的一个转换脚本.

引理 1 假设有字符串序列 $A=<A_1,A_2,\cdots A_n>$,$B=<B_1.B_2\cdots B_m>$,则 $|m-n|\leqslant D(A,B)\leqslant m+n-2\mathrm{MaxSub}(A,B)$

证明 假设 A 和 B 的最长公共顺序子序列为 C,我们可以通过如下插入/删除操作序列将 A 变为 B:首先通过 $m-\mathrm{MaxSub}(A,B)$次删除操作删除 A 中所有不包含在 C 中的字符串,然后通过 $n-\mathrm{MaxSub}(A,B)$次插入操作在相应位置插入 B 中所有不包含在 C 中的字符串. 因此有:$D(A,B)\leqslant m+n-2\mathrm{MaxSub}(A,B)$. 又每次插入/删除操作将序列的长度改变1,因此 $n-D(A,B)\leqslant m\leqslant n+D(A,B)$,即 $|m-n|\leqslant D(A,B)$.

定理 1 对于两条路径 $p1,p2$,其长度分别为 m 和 n,如果 $\mathrm{PathSim}(p1,p2,\delta,\theta)=\mathrm{true}$,则 $|m-n|\leqslant(1-\theta)(m+n)$

证明 令 $A=\mathrm{LS}(p1)$,$B=\mathrm{LS}(p2)$,则 A 和 B 的长度分别为 m 和 n,由算法1,$2\mathrm{MaxSub}(A,B)/(m+n)\geqslant\theta$,因此由引理1,$|m-n|\leqslant D(A,B)\leqslant(m+n)(1-2\mathrm{MaxSub}(A,B)/(m+n))\leqslant(1-\theta)(m+n)$.

根据定理1,假设查询 T 中的完整路径长度在 l 和 h 之间,令 $\min=\lceil l\theta/(2-\theta)\rceil$,$\max=\lfloor(2-\theta)h/\theta\rfloor$,其中上括号和下括号分别表示上取整和下取整,则不难推出我们只需要提取图 G 中长度在[min,max]之间的路径. 长度在此区间之外的路径不可能与 T 中的任何全路径相似. 路径提取算法可以利用动态编程的方法来完成,即:首先检查 G 中每一个值节点的每一条入边得到所有长度为1的完整路径,对于每条长度为 k 的完整路径,将其起始边的头节点的每条入边分别与其拼接即得到所有长度为 $k+1$ 的路径($1\leqslant k<\max$).

3.2 相似性连接

设路径提取步骤得到的完整路径集合为 R,令 Z 为 T 中所有全路径的集合,对于 Z 中的每一条路径,我们需检查其与 R 中每一条路径的相似性. 即将 R 与 Z 进行相似性连接. 这个过程可以利用关系数据库系统完成.

我们将集合 R 中的路径按起始标签的不同进行分类,并对每一类建立一个关系表来存储该类中的所有路径,表名定为该类中路径的起始标签,表结构为(pid, oid, path,len,text),表中 pid 是标识路径的关键字,oid 是路径起始节点的标识符,len 是路径长度,text 是该路径尾节点的文本值,path 是代表该路径标签序列的字符串:设路径的标签序列为$<A_1,A_2,\cdots A_n>$,则 path="$A_1\#A_2\cdots\#A_n$"(即由各个标签中间加上某个特殊字符'#'拼接而成). 相应地,建立一个表 Q(pid,path,len,text)来存储集合 Z 中的所有路径,Q 中字段含义类似,惟一不同的是 Q 中不设 oid 字段,因为 Q 中存储的所有路径的起始节点均为 root(T).

根据以上所建的关系表,假设查询 T 的根边的标签为 S,根据2.1节的判断路径相似算法,可以看出 Q 中的路径只可能与表 S 中的路径相似. 因此可以通过如下的 SQL 语句进行相似性连接:

```
Q1:  SELECT S.oid, S.pid, Q.pid  FROM  S, Q
     WHERE CheckPathSim(S.path, Q.path, S.len, Q.len, S.text, Q.text, δ, θ);
```

其中，CheckPathSim 是用户自定义函数，根据算法 1 来判断两条完整路径的相似性. 该 SQL 语句在关系数据库中的执行顺序是：先得到 S 和 Q 的笛卡尔乘积，然后对其中每一对路径计算 CheckPathSim. 由于 CheckPathSim 函数执行的时间复杂度至少为两条路径的长度的乘积，因此，在 S 和 Q 中存储的路径较多时，Q1 执行的时间复杂度较高. 定理 1 提出了两条路径相似的必要条件，因此可以利用定理 1 作为一个“过滤”条件. 如果两条路径相似，则其标签序列中应该具有较多的公共连续子序列，这种连续子序列称为“q 窗口”[12].

定义 6　假设 p 是 G 中的路径，q 为自然数，设 $\mathrm{LS}(p)=<A_1,A_2,\cdots A_n>$，令 Λ 是某个非 G 中标签的字符串常量，对任意整数 $j\notin[1,n]$，定义 $A_j=\Lambda$. 则序列 $\sigma=<A_{i-q+1},A_{i-q+2},\cdots A_i>(1\leqslant i\leqslant n+q-1)$ 称为 p 的一个 q 窗口. i 称为窗口位置，记为 $\mathrm{pos}(\sigma,p)$. 对于路径 $p1$ 和 $p2$，如果通过某个 A 到 B 的转换脚本能将 A 中的 q 窗口 $\sigma1$ 转换为 B 中相同的窗口 $\sigma2$，则称 $\sigma1$ 和 $\sigma2$ 是对应的 q 窗口. 记 $\mathrm{COMM}(p1,p2,q)$ 为 $p1$ 和 $p2$ 所有对应 q 窗口对的个数. 两条路径的对应 q 窗口可能具有不同的窗口位置. 然而，如果该两条路径相似，则对应 q 窗口的窗口位置应满足如下约束：

定理 2　假设完整路径 $p1$、$p2$ 长度分别为 m、n，$\sigma1$、$\sigma2$ 是 $p1$、$p2$ 对应的 q 窗口，$\mathrm{pathsim}(p1,p2,\delta,\theta)=\mathrm{true}$，$\mathrm{pos}(\sigma1,p1)=i$，$\mathrm{pos}(\sigma2,p2)=j$，则 $|i-j|\leqslant(1-\theta)(m+n)$

证明　令 $A=\mathrm{LS}(p1)$，$B=\mathrm{LS}(p2)$，由 A 可以经过 $D(A,B)$ 次插入/删除操作序列变为 B，而每一次插入/删除操作最多将一个 q 窗口的位置改变 1. 因此，对应 q 窗口的位置差异不超过 $D(A,B)$，即 $|i-j|\leqslant D(A,B)$. 再由定理 1 得证.

两条相似路径对应的 q 窗口除了满足定理 2 所指出的位置约束之外，还应满足如下的个数约束：

定理 3　对于两条完整路径 $p1$ 和 $p2$，其长度分别为 m 和 n，如果 $\mathrm{PathSim}(p1,p2,\delta,\theta)=\mathrm{true}$，则 $\mathrm{COMM}(p1,p2,q)\geqslant\max(m,n)+q-1-(1-\theta)(m+n)q$.

证明　令 $A=\mathrm{LS}(p1)$，$B=\mathrm{LS}(p2)$，由于可以经过 $D(A,B)$ 次插入/删除操作将 A 变为 B，而每一次插入/删除操作最多改变 q 个 q 窗口，因此 $D(A,B)$ 次插入/删除操作最多改变 $qD(A,B)$ 个 q 窗口. 而由定义 6，$p1$ 共有 $m+q-1$ 个 q 窗口，因此，$\mathrm{COMM}(p1,p2,q)\geqslant m+q-1-D(A,B)q$，又由定理 1 的证明可得 $D(A,B)\leqslant(1-\theta)(m+n)$，故 $\mathrm{COMM}(p1,p2,q)\geqslant m+q-1-(1-\theta)(m+n)q$. 又由于 $D(A,B)=D(B,A)$，故同理可证 $\mathrm{COMM}(p1,p2,q)\geqslant n+q-1-(1-\theta)(m+n)q$.

至此，我们已经找到了两条路径相似的 3 个必要条件（即定理 1、2、3）. 利用这些必要条件，我们可以在 SQL 语句中首先“过滤”掉表 S 和 Q 的笛卡尔乘积中不可能相似的路径对. 为此，除 S 和 Q 之外，建立辅助表 Sqwin 和 Qqwin 来存放 S 和 Q 中路径的 q 窗口，其表结构均为（pid，pos，qwin，len），对 S 中的每条路径 p，在 Sqwin 中记录其所有的 q 窗口，其中 pid 是外键，指向 S. pid，pos 是 q 窗口的窗口位置，qwin 是代表 q 窗口的字符串（如同 S. path 一样由标签拼接而成），len 是路径 p 的长度. 注意 Sqwin 表可以在查询到来之前预先建立. 同理，利用 Qqwin 来记录 Q 中路径的所有 q 窗口. 生成辅助表之后，我们可以得到如下的相似性连接查询语句 $Q2$：

```
Q2: SELECT    S.oid, S.pid, Q.pid
    FROM      S,Q,Sqwin,Qqwin
```

```
WHERE    S. pid=Sqwin. pid  AND Sqwin. qwin=Qqwin. qwin  AND Qqwin. pid=Q. pid
         AND |Sqwin. pos-Qqwin. pos|≤(1−θ)(Sqwin. len+Qqwin. len)
         AND |S. len−Q. len|≤(1−θ)(S. len+Q. len)
GROUP BY S. oid, S. pid, Q. pid, S. len, Q. len, S. path, Q. path
HAVING   COUNT(*)≥Max(S. len,Q. len)+q−1−(1−θ)(S. len+Q. len)q
         AND CheckPathSim(S. path, Q. path, S. len, Q. len, S. text, Q. text, δ, θ)
```

在 $Q2$ 中,首先,GROUP BY 子句将 FROM 子句的表连接结果按照每一对 S 和 Q 的路径分组,接着 Where 子句根据定理 1 和定理 2 滤去不可能相似的路径对以及不可能对应的 q^- 窗口,而后 Having 子句中的 COUNT(*)子句滤去不满足定理 3 的路径对. 通过这些过滤后,最后只需在 Having 子句中对较少的路径对计算复杂度较高的 CheckPathSim 函数,从而提高了查询效率.

3.3 得到查询结果

根据上一步的相似性连接,可以得到所有的相似路径对,每个路径对中的两条路径分别是数据源 G 中的完整路径和查询 T 中的全路径. 我们建立表 U=(oid, pid1, pid2)来存储这些相似路径对,其中 pid1,oid 是 G 中路径的标识符和其起始节点的标识符,pid2 是 T 中的路径的标识符. 为实现近似查询功能,根据定义 4,设 T 中有 h 条全路径,则如下语句返回近似查询的结果:

```
Q3: SELECT   oid, COUNT(DISTINCT pid2)   FROM  U
    GROUP BY oid
    HAVING   COUNT(DISTINCT pid2)> λh
    ORDER BY COUNT(DISTINCT pid2) DESC
```

在以上查询语句中,我们将 U 中元组按照节点标识符 oid 分组,对每一个分组中的节点,根据定义 4,COUNT(DISTINCT pid2)若满足 HAVING 子句的条件,则为该节点与查询 T 的相似度. 最后,ORDER BY 子句保证了按照相似度降序排列将结果返回给用户. 至此我们可以看出,在整个相似查询的过程中,我们充分利用了关系数据库的查询功能.

3.4 试验

我们采用一台 Athlon 1G,256M 内存的机器来进行试验. 数据库系统采用 oracle 9i. 操作系统为 windows xp,并用 java 语言和 jdom 接口来实现整个系统. 我们根据 DTD 文档 Gedml. dtd[3],利用 IBM XML generator[13]生成图结构的 XML 数据作为数据源. Gedml. dtd 在文献[3]中被用来生成"高度不规则"的图结构 XML 文档. 我们共生成 3 个大小不同的 XML 文档. 这些文档的各个参数如表 1.

表 1 数据集参数

Tab. 1 Parameters of data set

文档	节点数	IDREF 连接数	边数	路径数(T1)	最大表记录数(T1)	路径数(T2)	最大表记录数(T2)
Gedml_1	825	42	866	3 983	951	34 174	6 349
Gedml_2	2 114	86	2 199	9 581	2 065	65 905	11 570
Gedml_3	4 261	152	4 412	20 140	6 482	184 388	29 063

在 Gedml. dtd 的基础上,我们通过引入不同的路径嵌套结构和文本值生成两个树查询

$T1$ 和 $T2$，其中，$T1$ 共有 50 条全路径，其路径长度从 1 到 5；$T2$ 共有 100 条全路径，其路径的长度从 1 到 10. 试验中取阈值 $\delta=3$，$\theta=0.8$，$\lambda=0.8$，$q=2$. 从各个文档中提取的路径数见表 1，其路径提取时间均在 1 秒以内. 提取的路径用 3.2 节的建表方法根据起始标签分类并存放在不同的关系表中，这些表的最大表记录数见表 1. 对每个文档，我们分别测试其对于 $T1$ 和 $T2$ 的关系查询 $Q1$ 和 $Q2$ 的时间（$T1$ 和 $T2$ 的根边标签设置为与相应最大表中路径的起始标签相同，以获得最坏情况下的查询执行时间）. 在所有的情况下，$Q3$ 的时间均在几百毫秒以内，因此相对于 $Q1$ 和 $Q2$ 可以忽略不计（$Q1$、$Q2$、$Q3$ 的定义如 3.2 节）. 图 2 给出了各个查询的响应时间. 可以看出，当数据源文档的大小变大及 IDREF 数变多时，路径数也增多. 然而通过将抽取的路径按照起始标签分类，最大表的记录数仍然在相对较小的值，从而减少了查询开销. 当最大表记录数增多或查询树的大小变大时，$Q1$ 和 $Q2$ 的查询时间均变大，这是可以理解的，因为此时由于路径数的增多，相似性连接的代价也较大. 而对于每一个数据源文档和查询 $T1$、$T2$，$Q1$ 的查询时间都远远大于 $Q2$ 的查询时间. 这证明了 3.2 节中通过引入过滤条件来提高查询效率的策略的有效性.

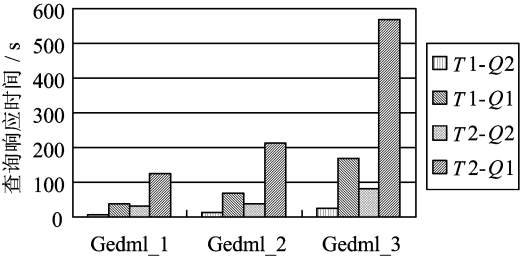

图 2　查询响应时间（秒）

Fig. 2　Query response time(s)

4　结束语

本文提出一种利用关系数据库系统对于一般图结构的半结构化数据进行近似查询的方法，根据嵌套结构和尾节点文本值内容的相似性得到两条路径的相似性度量，并根据路径相似性得到节点的相似性度量. 为进行近似查询，首先提取出数据源和查询中的路径集合，将其存储到关系数据库中后利用 SQL 语句进行路径的相似性连接并最终得到查询结果. 在相似性连接的过程中利用若干相似必要条件来提高查询效率. 实验结果证明了有效性.

参　考　文　献

[1] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg. A Comparison of string distance metrics for name-matching Tasks [A]. Proceedings of IJCAI, 2003.

[2] Durban R, Eddy S R, Krogh A, *et al*. Biological sequence analysis-probabilistic models of proteins and nucleic acids [M]. Cambridge: Cambridge University Press, 1998.

[3] CHUNG Chin-wan, MIN Jun-ki, Kyuseok Shim. APEX: an adaptive path index for XML data [A]. SIGMOD Conference [C], 2002: 121-132.

[4] ZHANG Chun, Jeffrey F Naughton, David J DeWitt, *et al*. On supporting containment queries in relational database management systems [A]. SIGMOD Conference 2001 [C].

[5] LI Quan-zhong, Bongki Moon. Indexing and querying XML data for regular path expressions [A]. VLDB 2001 [C]: 361-370.

[6] Igor Tatarinov, Stratis Viglas, Kevin S Beyer, *et al*. Storing and querying ordered XML using a relational database system [A]. SIGMOD Conference [C], 2002: 204-215.

[7] Jayavel Shanmugasundaram, Eugene J, She-

kita, Jerry Kiernan, *et al*. A general techniques for querying XML documents using a relational database system[J]. SIGMOD Record 30(3): 20-26 (2001).

[8] Torsten Schlieder. Schema-driven evaluation of approximate tree-pattern queries[A]. 8th International Conference on Extending Database Technology Proceedings[C]. Prague, Czech Republic, March 25-27: 514-532.

[9] Sihem Amer-Yahia, SungRan Cho, Divesh Srivastava. Tree pattern relaxation[A]. 8th International Conference on Extending Database Technology Proceedings[C]. Prague, Czech Republic, March 25-27: 496-513.

[10] Daniel S Hirschberg. A linear space algorithm for computing maximal common subsequences[J]. Communications of the ACM, 1975,18(6): 341-343 .

[11] Alvaro E Monge, Charles Elkan. The field matching problem: algorithms and applications[A]. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)[C]: 267-270.

[12] Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Lauri Pietarinen, Divesh Srivastava: Using q-grams in a DBMS for Approximate String Processing. IEEE Data Engineering Bulletin 24(4): 28-34 (2001).

[13] http://www. alphaworks. ibm. com/tech/xmlgenerator[DB/OL].

Approximate Querying of Semistructured Data Using a Relational Database System

HAN Kai, YUE Li-hua, GONG Yu-chang.

(*Department of Computer Science and Technology, USTC, Hefei* 230026, *China*)

Abstract: An approach to approximate querying of general graph structured semistructured data is proposed based on a relational database system. A similarity measure for paths based on nesting structures and text values was brought out, from which the similarity between the target query node and a data source node was derived. To get the source nodes similar to the target query node, firstly the paths whose lengths were within an interval were extracted from the data source, then a similarity join process between them and the query paths was carried out using a relational database system. Finally the query result nodes were returned in a descend order of their similarity to the target query node. To make the similarity join process more efficient, the concept of q-windows was introduced and several necessary conditions were used to decrease the time of calculating costly similarity function. The experiments prove the effectiveness of the approach.

Key words: semistructured data; graph structured; approximate querying; similarity measure; similarity join; RDBMS