### 加密的XML文档上有效的树模式查询

#### 摘要

外包XML文档是一项具有挑战性的任务，因为它加密了文档，同时仍然需要高效的查询处理。过去关于此主题的方法要么泄漏结构信息，要么无法支持对XML节点内容有约束的搜索。此外，它们采用了一个过滤和重新定义的框架，要求用户从查询结果中删除误报。为了解决这些问题，我们提出了一种有效评估加密XML文档树模式查询（TPQ）的解决方案。我们创建了一个域层次结构，这样每个XML文档都可以嵌入其中。通过为层次结构中的每个节点分配一个位置，我们为每个文档创建一个向量，该向量对文档的结构和文本信息进行编码。类似地，还为TPQ创建向量。然后，减少TPQ和文档之间的匹配以计算它们的矢量之间的距离。为了保护隐私，在外包之前对这些载体进行加密。为了提高匹配效率，我们使用k-d树将矢量划分为非重叠子集，以便尽可能早地修剪不匹配的文档。广泛的评估表明，我们的解决方案非常有效，可扩展到大型数据集。

1. 介绍

由于柯达于1989年与IBM，DEC和Businessland [10]签订了10亿美元的合同，将其信息系统外包，因此数据外包引起了广泛关注。 外包对数据所有者是有益的。 它有助于节省构建和维护私有数据库系统的成本，从而使数据所有者能够专注于其核心竞争力。 最近，由于网络和计算技术的进步，云已经成为一种能够以相对低的价格提供可靠和灵活的数据访问服务的技术。 因此，数据所有者甚至更有可能外包他们的数据。然而，尽管具有所有吸引人的特征，但是将数据移动到云服务器可能危及个人隐私，因为数据可能包含敏感信息（例如，医疗记录）。 为了解决这些问题，数据通常在外包之前进行加密，这使得高效的查询处理非常具有挑战性。 在过去十年中，对加密关系数据的查询处理主题进行了大量研究[11,12,22,19]。 然而，过去的研究并没有很好地解决如何搜索加密的XML文档。

XML文档在层次结构中组织数据，并通过用户定义的标记描述数据元素之间的语义关系。 因此，XML文档包含结构和文本信息。 为了实现强大的隐私，重要的是要加密这两种类型的信息。 这使得为关系数据开发的现有外包技术[11,12,22]不适用于XML文档，因为它们不支持搜索加密的结构信息。

过去的研究[23,6]研究了对加密XML文档的查询处理。 然而，这种方法会泄漏结构信息[23]或无法支持对节点内容有约束的搜索[6]（详见第8节）。 此外，他们采用了一个过滤和重新定义框架，其中云服务提供商返回搜索结果的超集。 因此，用户必须通过另外的后处理步骤从返回的结果中移除错误的候选者。 为了解决这些问题，在本文中，我们考虑了加密XML文档的树模式查询（TPQ）[7,18]。 TPQ是XQuery的核心操作[3]，它现在是XML查询处理语言的事实标准。 具体而言，TPQ是一棵树，由标记的节点和谓词组成，用于指定节点上的约束。 为了支持TPQ的有效评估，我们为每个文档构建一个向量（即索引），如下所示。

我们假设存在域层次结构，该层次结构由数据集中所有XML文档的文档类型定义（DTD）组成。 每个文档都嵌入在层次结构中。通过为层次结构中的每个节点分配一个位置，我们为每个文档创建一个向量（类似数组，将层次结构中的所有节点存储到一个比特数组中，文档中存在该节点则相应数组位置值为’1’，否则为’0’），该向量对文档的结构和文本信息进行编码。 同样，我们也为TPQ创建了一个向量。 因此，TPQ与XML文档的匹配被减少到计算它们的矢量表示之间的距离。 然后，我们通过ASPE [24]加密矢量，确保安全性，同时支持加密矢量的距离比较（即KNN搜索）。此外，为了提高搜索效率并因此很好地扩展大型数据集，我们采用k-d树将向量划分为非重叠子集，以便尽可能早地修剪TPQ的不可匹配XML文档。

与现有方法[23,6]相比，我们的解决方案既保护XML文档的结构和内容，又支持对它们的查询处理。 我们广泛的实验结果也表明我们的解决方案非常有效。 此外，我们不采用过滤和重定义框架[23,6]。 我们的解决方案返回精确的查询结果，没有误报或漏报。

本文的其余部分安排如下。 下一节首先提出问题所在。 然后，我们在第3节中介绍了我们的方法，并在第4节中提高了它的效率。我们在第5节中讨论了我们的方法，并在第6节中分析了它的隐私。第7节报告了我们的实验结果。 最后，我们将在第8节中回顾相关工作，并在第9节中完成我们的工作。

1. 提出问题

我们将XML文档建模为带标签的有根树。 树中的每个节点都有一个名称。 节点是元素或属性。 叶元素和属性可以具有内容，其是字符串或数值。

在我们的设置中有三个角色：数据用户Alice，数据所有者Bob和云服务器Charlie。 我们假设Bob有一组包含敏感信息的XML文档，并且他希望将它们外包给云服务器Charlie。 为确保保密性，数据以加密形式存储在Charlie中。 授权数据用户Alice应该能够通过TPQ选择性地从Charlie检索XML文档（参见下面的定义2）。为了查询隐私，TPQ也是加密的。服务器应该能够评估加密的TPQ，并返回满足TPQ中指定的约束的加密文档。一旦接收到查询结果，Alice就会解密它们并以明文形式获得所需的文档。此外，我们假设Charlie是半诚实的，即，他严格遵循查询处理协议，因为它被定义但他可能尝试从查询评估中推断私人信息。

树模式查询：XQuery [3]是XML查询处理语言的当前事实标准。由于其复杂性，完全支持它超出了我们的工作范围。相反，我们考虑树模式查询（TPQ）[7]，它是其核心运算之一。TPQ具有广泛的应用 - 除了XML查询处理外，它还可以应用于Web数据管理和选择性数据传播。TPQ与XPath的语义兼容[2]。其正式定义如下。

定义1（树模式）：树模式是树，使得1）其每个节点具有名称，以及2）其每条边是表示父子（PC）关系的单边或表示祖先（AD）关系的双边。

定义2（树模式查询）：树模式查询是一对（Qt，Qc），其中Qt是树模式，Qc是在Qt的节点上定义的谓词的布尔组合。

我们支持两种谓词。 第一个是[x op α]形式的内容谓词，其中x表示节点的内容，op∈{>，≥，<，≤，=}，α是一个值。 它选择内容值满足谓词的节点。 与在XPath中一样，我们通过节点名称表示节点的内容。 第二个是[position() = m]形式的位置谓词。它选择当前上下文节点的第m个子节点。

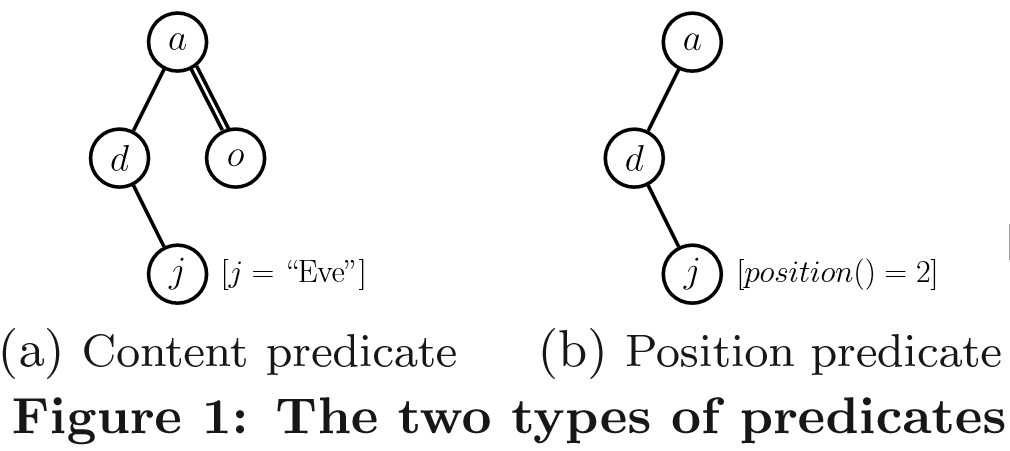
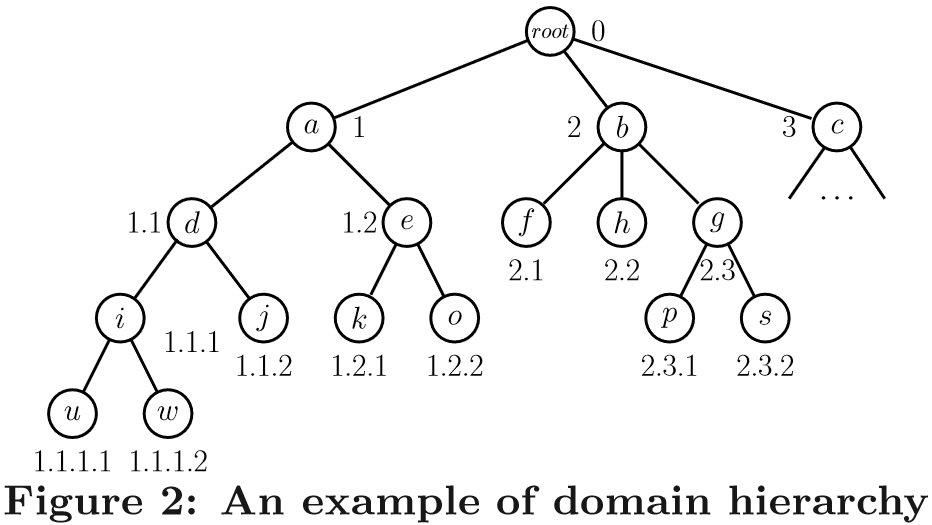


图1（a）显示了TPQ的一个例子，其中a和d（d和j）通过PC关系连接，a和o通过AD关系连接，并且j的内容需要等于 “Eve”。 由于TPQ遵循XPath的语义，因此示例TPQ也等于以下两个XPath查询的组合：Path1 = / child :: a / child :: d / child :: j [j =“Eve”]和Path2 = / child :: a / descendant :: o。 图1（b）显示了具有位置谓词的TPQ的示例，其中应选择具有节点名称j的d的第二个子节点。

1. 解决方案

在本节中，我们将介绍加密XML文档中评估TPQ的方法。 在3.1节中，我们将首先介绍将XML文档（和TPQ）编码为向量的策略。 然后，在3.2节中，我们根据我们的特定要求调整加密方案。 根据前两节，第3.3节给出了我们方法的细节。



3.1将XML文档编码为向量

我们为XML文档创建向量。由于这些向量编码文档的结构和文本信息，因此它们可以被视为用于高效查询处理的索引。为了生成向量，我们假设数据库中的所有XML文档都是根据其文档类型定义（DTD）构建的。我们在所有DTD上添加一个根节点，以构建域层次结构。考虑图2中的示例，其中以节点a，b和c为根的子树表示三个DTD。 XML文档可以被视为域层次结构中的子集，或者更确切地说，是域层次结构的嵌入（请参阅下一个定义）。

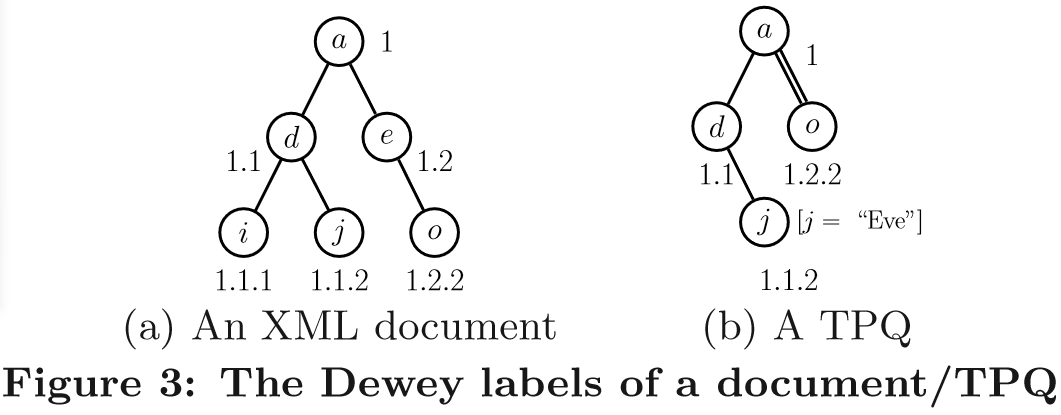
定义3（树嵌入[13]）： 设T和T'分别是具有多组节点V和V'的树。内射函数f：V→V'是一个T到T'的嵌入，如果对于所有节点x，y∈V：

•label（f（x））= label（x），其中label（f（x））和label（x）分别是树T'和T中f（x）和x的标签，

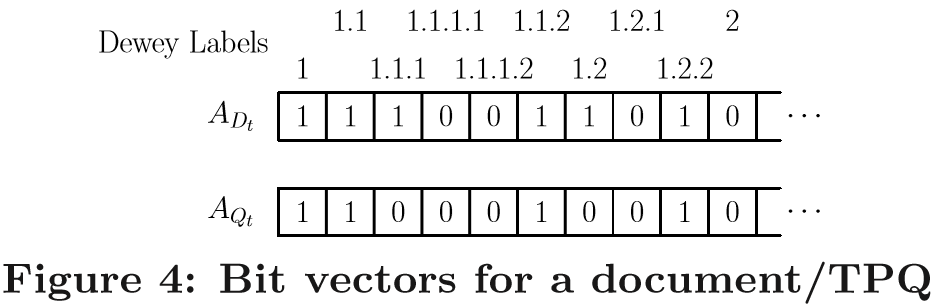
•f（x）是T'中f（y）的后代，当且仅当x是T中y的后代时。

根据DTD，允许一些元素（例如，书籍的作者）在XML文档中多次出现。 为确保每个XML文档都是域层次结构中的嵌入，我们复制域层次结构中的某些节点。 作为示例，请考虑图2.假设节点j表示书籍的作者姓名，并且M是数据集1中书籍的最大作者数。 然后，我们在域层次结构中的节点d下复制j M次（有关更详细的讨论，请参阅第5节）。

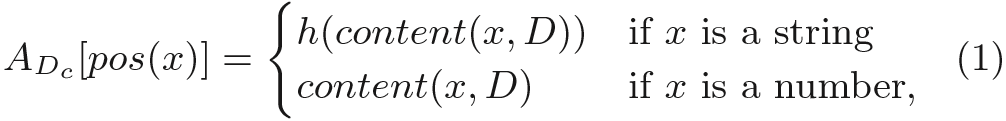
一旦域层次结构准备就绪，**我们利用Dewey labeling方案[25]来标记域层次结构。** 设p和c分别是Dewey标记a1.a2 ... am和b1.b2 ... bn的两个节点。 如果p是c的父，则ai = bi（i = 1,2，...，m），n = m +1。 节点c的最后一个组件bn表示其兄弟之间的c的本地顺序。 考虑图2中的示例，其中根节点下的每个子树都标有Dewey标签。 基于域层次结构上的这种标记，可以相应地标记XML文档。 图3（a）显示了一个示例XML文档及其Dewey标签。 TPQ可以被视为树，因此也可以被标记。 图3（b）中的示例是图1（a）中TPQ的一种可能标记。 TPQ可能在域层次结构中具有多个嵌入，因此它可能具有多个标记（有关更多详细信息，请参阅第5节）。



域层次结构中的所有Dewey标签（表示为U）可以视为通用集。相应地，XML文档（或TPQ）中的Dewey标签集可以被视为U的子集。如果我们进一步在U中对标签进行排序，那么我们可以将文档（或TPQ）的结构转换为二进制向量。 令X为文档（或TPQ）的Dewey标签集合，使得X⊆U。令pos（x）为标签x在U中的序号。我们用以下方式为X构造一个位向量A：如果x∈X则A [pos( x )] = 1，否则A [pos( x )] = 0。 考虑图4中的示例，ADt和AQt分别是图3中XML文档和TPQ的结构信息的位向量。



位向量表示有助于结构比较。给定文档和 TPQ的位向量，分别表示为ADt和AQt，我们可以很容易地检查它们的结构是否匹配。特别是，如果AQt和ADt之间的内积等于AQt中的1的数，那么可以断定该文档与查询匹配。 此外，我们还可以将XML文档的节点值编码为向量，以支持基于内容的匹配。 给定XML文档D，我们按如下方式创建其文本向量ADc。 对于任何节点x∈U，如果x∈D且x在D中具有值，则ADc [pos( x )]为：



其中h是从{0,1} \*到{0,1} l的加密散列函数，而content（x，D）表示D中节点x的值。另一方面，如果x /∈D，或x ∈D但它没有任何值，则将与公式1中定义的任何值都不同的特殊整数ω，分配给ADc [pos( x )]。 作为一个简单的解决方案，我们可以移动所有数值，使它们都大于或等于1，并设置ω= 0。注意TPQ中谓词的编码与上述不同（参见第3.3节中的查询结构）。

示例1. 考虑图3（a）中的XML文档D。假设节点j表示名字，其在D中的值等于“Adam”。此外，假设标签域中的j的顺序是6。然后，D的文本向量的第六维被设置为h（Adam），即，ADc [6] = h（“Adam”）。

3.2 ASPE（这一节的公式较多直接看原文）

**我们在这里采用的加密策略基于非对称标量产品保留加密（ASPE）方案，该方案在[24]中提出用于云上的高效安全最近邻搜索。**特别地，假设P1和P2是两个数据点，并且Q是欧几里德空间中的查询点。 如果P1，P2和Q由ASPE加密，则第三方将不会知道数据点和查询的值，但它仍然可以确定P1是否比P2更接近Q，ASPE的构建块简要概括如下：

•密钥key：两个（n + 1）×（n + 1）的可逆矩阵M1和M2，以及长度为n + 1的二进制串S.

•数据加密函数E1：设P是n维数据点。将P扩展到P =( P^T, -0.5||P||2 ) ^ T。创建（Pa，Pb），使得：1）如果S [i] = 1，则设置Pa [i] =ρi并且Pb [i] = P [i] - ρi，其中ρi是随机数，并且2）如果S [i] = 0，则设置Pa [i] = Pb [i] = P [i]。 P的加密是E1（P）= [（MT 1 Pa）T，（MT 2 Pb）T] T.

•查询加密函数E2：设Q为查询点。将Q扩展到b Q = r（QT，1）T，其中r是正随机数。创建（Qa，Qb），使得：1）如果S [i] = 1，则设置Qa [i] = Qb [i] = b Q [i]，以及2）如果S [i] = 0，则设置Qa [i] =σi和Qb [i] = b Q [i] - σi，其中σi是随机数。 Q的加密是E2（Q）= [（M-1 1 Qa）T，（M-1 2 Qb）T] T.

•比较功能Comp：令E1（P1），E1（P2）和E2（Q）分别是两个点P1和P2以及查询Q的加密。为了检查P1是否比P2更接近Q，该功能检查（E1（P1）-E1（P2））⊙E2（Q）> 0，其中⊙是内积。

在下文中，我们简要讨论协议的正确性。 正式证明可以在[24]中找到。

事实1 Fact 1.

P ⊙b Q = Pa ⊙ Qa + Pb ⊙ Qb

Fact 2.

(P1 − P2) ⊙b Q = 0.5r(d2(P2,Q) − d2(P1,Q)),

其中函数d测量两点之间的欧几里德距离。 基于这两个事实，我们得到

(E1(P1) − E1(P2)) ⊙ E2(Q) =......, (2)

这意味着如果（E1（P1）-E1（P2））⊙E2（Q）> 0，则P1比P2更接近Q. 为了清楚地呈现，在下文中，我们通过以下等式表示比较函数：

Comp(E1(P1),E1(P2),E2(Q)) =......

Wong等人[24]表明ASPE的安全性大致等于具有n位密钥的对称加密方案。 为了确保足够的安全性，它设置n≥80。如果数据点的尺寸小于80，则会添加一些额外的尺寸（详见[24]）。

在我们的工作中，我们利用ASPE的特征来支持对一维点P的特定维度的查询。特别地，假设P [λ]是P的维数，并且α是与p[λ]的含量进行比较的数值。我们首先生成以下两个n维向量：

Q1λ = (γ1,γ2,··· ,γλ−1,α − s,γλ+1,··· ,γn)

Q2λ = (γ1,γ2,··· ,γλ−1,α + s,rλ+1,··· ,γn),

其中s和γi（i=1,2，…，λ−1，λ+1，…，n）是随机选择的正数。

然后我们使用数据加密函数E1来加密Q1λ和Q2λ，并使用查询加密函数E2来加密P.这样的结构使得能够通过ASPE函数比较查询内容α与P [λ] Comp如下：

公式（3）

3.3加密XML文档上的私有TPQ

给定TPQ，可以单独评估其树模式和谓词。 特别是，令ADt和ADc分别为XML文档D的结构和文本向量。假设Q =（Qt，Qc）是一个TPQ，AQt是其结构编码而成的位向量。 然后，ADt和AQt之间的内积可以确定D是否与结构相匹配。 设[x，op，α]为Qc中的谓词。 然后，将ADc和[x，op，α]作为输入的评估（即公式3）可以确定D是否满足谓词条件。 但是，在这种方法中，文档可能只匹配TPQ的树模式，但与TPQ的谓词不匹配。 在查询评估之后，云服务器会注意到这一点。 在某些情况下，由于对结构和内容的单独处理而导致的这种信息泄漏可能是不被期望的。

为了解决上述信息泄漏问题，我们制定了一种将结构和文本编码整合在一起的策略。 该策略由四个步骤组成：1）密钥生成，2）索引构建，3）查询构建和4）查询评估。

1. 生成密钥：设U是包含域层次结构中所有Dewey标签的通用集。 数据所有者Bob生成两个（| U | + 1）×（| U | + 1）可逆矩阵M1和M2，其中条目是合理的。 Bob还创建了一个（| U | + 1）位向量S.两个矩阵和S是密钥，它们与数据用户Alice共享，并将用于ASPE加密。
2. 构建索引：对于每个XML文档D，Bob首先创建两个向量ADt和ADc，它们分别对D的结构和节点内容进行编码。 然后，他将它们组合成单个矢量AD。 特别是，假设ADc [i]的位长最多为l（我们假设散列函数h的输出长度l长于数据库中任何数值的位长），其中i = 1,2，...，| U |。 Bob设置AD [i] = ADt [i]×2ℓ+ ADc [i]，其中i = 1,2，...，| U |。 以这种方式，AD封装文档D的结构和文本信息。最后，使用ASPE函数E2加密AD，并且将E2（AD）作为D的索引传送给Charlie。
3. 构建查询：设Q =（Qt，Qc）为TPQ，x为其中的节点。 Alice首先在域层次结构中嵌入Qt。 根据嵌入，假设标签域U中的x的顺序是pos（x）=λ。 然后，Alice根据Qc是否包含x上的内容谓词，以下列两种方式之一创建x的子查询。

情况1.在x上存在内容谓词[x，op，α]∈Qc，其中op∈{>，≥，<，≤，=}。在这种情况下，Alice计算¯α=2ℓ+α，并生成以下两个向量：

Q1λ=（γ1，γ2，...，γλ-1，α-s，γλ+ 1，...，γ| U |），

Q2λ=（γ1，γ2，...，γλ-1， ¯α+ s，γλ+ 1，...，γ| U |），

其中s和γi（i = 1,2，...，λ-1，λ+ 1，...，| U |）是正随机数。两个向量都由加密函数E1加密。最后，Alice创建了一个三元组（E1（Q1λ），E1（Q2λ），op）。当且仅当其索引使“AD [λ]op¯α”成立时，可以很容易地证明文档D满足谓词[x，op，α]。

情况2.节点x上没有内容谓词。在这种情况下，Alice计算¯α=2ℓ。然后，她也要生成Q1λ和Q2λ，并按照情况1对它们进行加密。根据索引构造，如果文档D包含节点x，则无论D是否具有x的内容，其索引AD中的第λ维必须大于或等于2ℓ，即AD [λ]≥2ℓ。因此，为了检查文档D是否包含节点x，Alice最终创建三元组（E1（Q1λ），E1（Q2λ），≥）。此外，我们可以看到案例2实际上是案例1的特例，它的内容谓词是[x，≥，0]。

TPQ可以包含多个节点。在每个节点上可能存在一个或多个子查询（例如，通过'∧'或'∨'连接的节点上的两个内容谓词）。 因此，Alice需要生成多个这样的三元组，每个子查询一个。之后，她以 联合正常形式（CNF）连接三元组，并将它们发送给查理。

1. 评估查询：给定加密的TPQ，其特征在于一组三元组（E1（Q1λ），E1（Q2λ），op），对于每个三元组，查理对其进行评估，并返回所有使得加密的查询评估为真的XML文档D。 如果满足以下条件，则三元组（E1（Q1λ），E1（Q2λ），op）对于文档D的评估结果为真：

Comp(E1(Q1λ),E1(Q2λ),E2(AD)) =......

1. 提高效率

在上一节中，我们提出了一个解决方案，该解决方案需要使用外包数据库中的每个XML文档来评估TPQ。 为了提高其计算效率，从而为大数据源很好地扩展，在本节中我们提出了两种优化技术。

第一种技术简单而有效。 它通过DTD对XML文档进行分区，这样在每个生成的分区中，所有文档都具有相同的DTD。 然后，它独立地处理每个分区，就像每个分区是一个独立的外包数据库一样。 特别是，对于每个分区，它独立地构建域层次结构，基于该层次结构构建分区中所有文档的安全索引。 给定TPQ，它首先找到分区，这样TPQ就可以嵌入到它们的域层次结构中（定义3）。 然后，仅使用这些分区中的XML文档评估TPQ。 显然，这提高了查询效率。 此外，上述优化技术还节省了安全索引的存储空间，因为这些索引是根据比以前更小的域层次结构构建的。

示例2.假设外包数据库中有三个DTD，图2是在这三个DTD上构建的域层次结构。 假设第一DTD对应于子树（a），即，以图2中的节点a为根的子树，第二DTD对应于子树（b），第三DTD对应于子树（c）。 然后，通过上述优化技术，外包数据库被分成三个分区，子树（a），子树（b）和子树（c）成为它们的域层次结构，每个层次结构专门用于一个分区。 现在考虑图3（b）中的TPQ。 在优化之前，需要使用整个数据库中的文档对其进行评估。 通过优化，只需要使用第一个分区中的文档进行评估，因为它只能嵌入子树（a）中。

**第二种看看就好**

第二种优化技术是在第一种优化技术之上的进一步改进。 根据DTD对XML文档进行分区后，分区中的文档数可能仍然很大。 为了进一步提高性能，我们为每个分区构建了一个k-d树，以便具有相似节点值的文档聚集在k-d树的同一个叶子中。

k-d树（k维树的简称）是用于索引k维数据点的二叉树。 它通过（k-1）维超平面递归地将数据空间划分为两个子空间。 树中的每个节点n与一个维度ax和ax轴上的分裂点sp相关联。 通常，sp是节点n中所有ax值的中值。 节点n的左子树中的所有数据点具有小于或等于sp的ax值，并且右子树中的所有数据点具有大于sp的ax值。 以这种方式，分裂点sp实际上设置垂直于ax轴的（k-1）维超平面。 超平面将节点n覆盖的空间分成两个（即，一个由左子树覆盖，另一个由右子树覆盖）。 通常，与循环相关联的维度以循环方式决定。 也就是说，级别dp的节点与第i维度相关联，其中i = dp mod（k）+ 1。

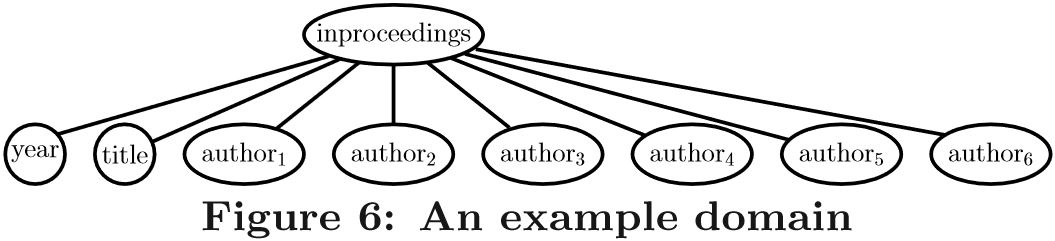
要为根据相同DTD创建的XML文档分区构建k-d树，我们从DTD中选择k个节点作为k维度。 节点值可以是数字或分类。 如果维度中的值是数字，我们可以直接使用它们。 对于分类值，我们需要将它们映射为整数。 有这种映射的各种方法。 在下文中，我们给出了一些可能的方法。 如果维度中的分类值在层次结构中以语义方式组织，我们可以通过层次结构的预先遍历遍历为每个值分配不同的整数，如[5]中所示。 如果这样的层次结构不可用，我们可以按照包含它们的XML文档数量的升序对分类值进行排序，然后为它们分配顺序整数。 此外，我们还可以考虑在分配整数之前概括分类值（例如，名称，如John，Jane和Jack，以字母'J'开头可以推广为'J \*'）。

示例3.考虑Inproceedings数据集中记录作者出版物的XML文档集。 在XML文档中，有一个“年”节点指示工作何时发布，并且还有一个“booktitle”节点，指示工作的发布位置。 我们可以分别将'year'和'booktitle'作为第一维和第二维，并在它们上构建一个k-d树。 因此，这里k = 2.如图5（a）所示，数据首先在'year'上划分，然后在'booktitle'上划分，然后在'year'上划分（即以循环方式）。 图5（b）显示了k-d树中的叶节点，它形成了Inproceedings数据集的XML文档的分区。

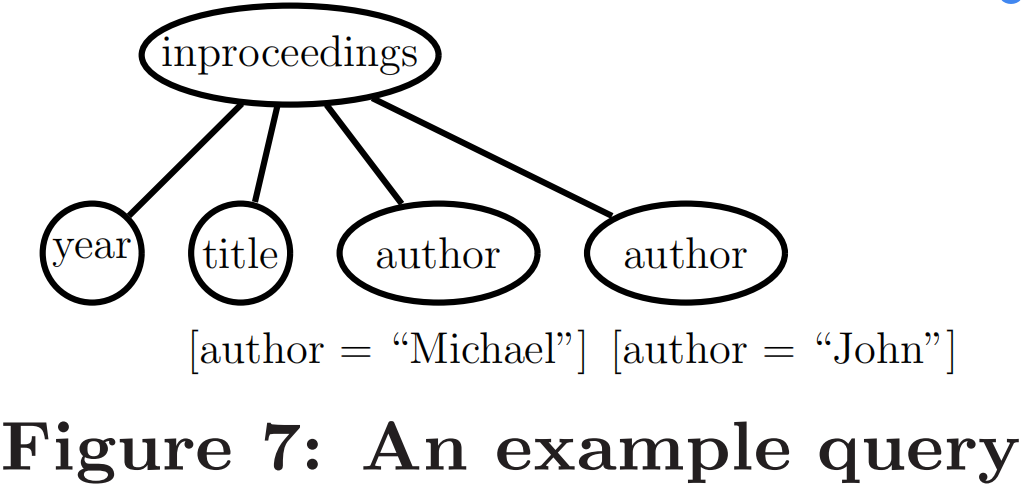
当k-d树可用时，需要仅在叶节点的子集（在k-d树中）中评估TPQ，其可能包含匹配的文档。 修剪所有剩余叶节点中的评估。 在例3中，如果Alice对1992年至1994年的SIGMOD论文感兴趣，那么只应评估分区R1中的文档。 结果，性能得到改善。

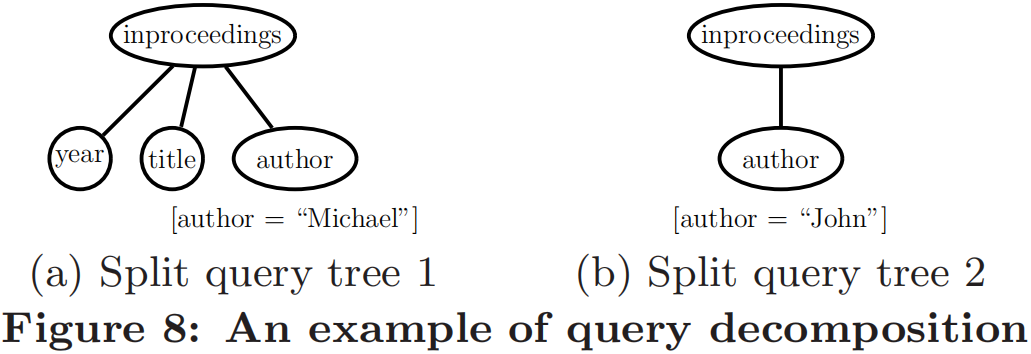
1. TPQ的多个嵌入式

给定具有Dewey标签的域层次结构，TPQ可能具有多个编码。 这种情况的发生主要是因为域层次结构中的多个节点可能具有相同的名称，尽管它们具有不同的Dewey标签。 考虑图6中的域层次结构。假设我们有一个带有单个节点作者的TPQ。 在这种情况下，TPQ可以映射到域层次结构中的6个作者节点中的任何一个。 因此，我们共有6个嵌入。 通常，这种查询的嵌入数量不大，即，它的上限是标签域的大小| U |。



在更复杂的情况下，TPQ具有多个节点，其中一些节点可以映射到域层次结构中具有相同名称的多个节点。如果是这种情况，则需要开发一些机制以避免为TPQ生成指数级的嵌入。再次考虑图6中的域层次结构。假设图7是TPQ。由于inproceedings元素具有6个作者的子元素，因此该查询具有C62 = 15种不同的嵌入。处理该问题的一种方法是数据用户给出特定位置（即，通过位置谓词[position（）= m]），这些查询节点应该被映射到该位置。例如，一旦用户要求TPQ中的两个作者应该映射到域层次结构中的前两个作者，那么查询只有一个嵌入。或者，用户还可以将原始查询分解为多个子TPQ。例如，用户可以将图7中的查询分解为图8中的两个子TPQ。图8（a）中的子TPQ具有6个嵌入，对应有6个内容谓词：πi = [authori, =,“Michael”]，其中i从1到6。类似地，图8（b）中的子TPQ也有6个嵌入，对应有6个内容谓词，即τj = [authorj , =,“John”]，其中j从1到6。现在第三方应该返回这些使得（Viπi）∧（Vjτj）评估为真的文档。请注意，在查询分解后现在我们总共只有12个嵌入。





就像TPQ一样，XML文档也可能在域层次结构中有多个嵌入。 但是，我们只需要为每个XML文档考虑一种可能的嵌入。 这足以支持TPQ和XML文档之间的匹配，因为我们已经枚举了TPQ的所有可能嵌入。 在我们的工作中，对于每个XML文档，我们只使用其最左边的嵌入。

1. 安全性分析

根据第3.3节，TPQ被分解为一组子查询，每个子查询用于TPQ中的一个节点。 更确切地说，对于具有内容谓词[x，op，α]的节点x，其中op∈{>，≥，<，≤，=}并且α是值，则为该谓词构造子查询（查询构造中的情况1）。 但是，当节点x没有内容谓词时，子查询也被构造为好像存在内容谓词[x，>，0]（查询构造中的情况2）。 因此，每个子查询可以推广为[x，op，α]的形式。 TPQ是这些谓词的子查询的组合。 因此，在下文中，为了简化分析，我们直接讨论这些谓词，而不是TPQ。

6.1无k-d树时的安全性

假设[x，op，α]是一个谓词，并且分配给x的位置是pos（x）=λ。 为了评估谓词所代表的查询，我们生成以下两个向量（参见第3.3节）：

Q1λ = (γ1,γ2,··· ,γλ−1,α − s,γλ+1,··· ,γ|U|),

Q2λ = (γ1,γ2,··· ,γλ−1,α + s,γλ+1,··· ,γ|U|),

其中s和γi（i = 1,2，...，λ-1，λ+ 1，...，| U |）是正随机数。 设AD是为XML文档生成的向量（即索引）。 通过检查AD是否与Q1λ和Q2λ接近，服务器可以决定AD [λ] =α。 这种比较允许服务器了解哪些外包文件满足公式AD [λ] =α，尽管服务器既不知道AD [λ]也不知道α，它们已经由ASPE加密。 我们注意到这种信息泄露是所谓的访问模式[8]，这是很难防止的，几乎所有有效的私有关键字搜索方案都泄露了这些信息。

我们在3.3节中的协议支持范围查询。 给定谓词[x，op，α]，其相关向量Q1λ和Q2λ以及XML文档的向量AD，服务器通过检查AD是否比Q2λ更接近Q1来知道AD [λ] <α。 因此，在评估对应于谓词[x，op，α]的子查询之后，服务器能够将所有外包的XML文档分成三个不相交的子集：1）x值等于α的文档，2）x的文档 值小于α，和3）x值大于α的文档。 但是，在后两个子集中的任何一个中，服务器都不能按x值的升序或降序对文档进行排序。

引理1：假设[x，op，α]是谓词，x的顺序是pos（x）=λ。 设AD1和AD2为两个数据点，使得AD1 [λ] <α和AD2 [λ] <α。 服务器无法确定AD1 [λ]> AD2 [λ]。

证明：设Q1λ和Q2λ为根据谓词生成的点。 注意，只有当d2（Q2λ，AD2） - d2（Q1λ，AD2）> d2（Q2λ，AD1） - d2（Q1λ，AD1）时，AD1 [λ]> AD2 [λ]。 但由于d2（Q2λ，AD2） - d2（Q1λ，AD2）和d2（Q2λ，AD1）-d2（Q1λ，AD1）受到不同随机值的保护，即（Q1λ - Q2λ）Td AD1 = 0.5r1（d2） （Q2λ，AD1） - d2（Q1λ，AD1））和（Q1λ-Q2λ）Td AD2 = 0.5r2（d2（Q2λ，AD2） - d2（Q1λ，AD2）），云服务器无法判断AD1[ λ]是否大于AD2 [λ]。

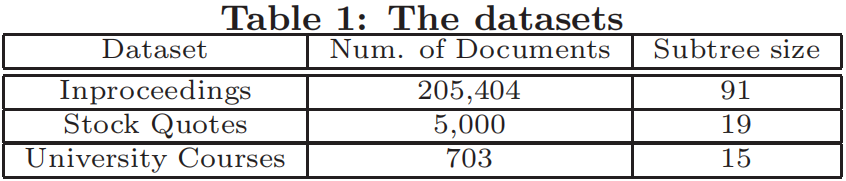
注意，在只需要相等匹配功能的场景中，实际上我们可以在将它们发送给服务器之前随机化E1（Q1λ）和E1（Q2λ）的顺序，以防止服务器知道AD [λ]是否小于α。 以这种方式，服务器通过检查AD [λ]是否与Q1λ和Q2λ处于相同的距离，仍然可以进行相等比较。 但是，由于服务器无法区分E1（Q1λ）和E1（Q2λ），因此无法判断AD [λ]是否小于α。

6.2有k-d树时的安全性

......

1. 实验报告

在本节中，我们对提出的方案进行了全面的实验评估。 我们使用3个XML数据集（表1）。 第一个数据集是Inproceedings，它是DBLP数据集的一个子集[15]。 DBLP数据集中元素节点Inproceedings下的每个子树代表一个出版物，该出版物出现在主要的计算机科学期刊或会议中。我们将每个这样的子树作为XML文档，这个数据集总共有205,404份文件。股票行情是随机生成的纳斯达克股票报价[16]，其中包含有关5,000种股票的信息。第三个数据集是大学课程[17]，它记录了里德学院网站的课程数据。 它包含703个文档。 我们的解决方案的原型是用Java实现的，实验是在Intel Core i7-2600 3.40GHz CPU机器上进行的，其中8G字节内存运行Linux 3.4.13。

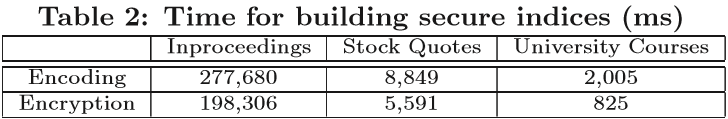


7.1构建索引

我们通过以下三个步骤为XML文档构建安全索引：1）构建域层次结构，2）为每个XML文档生成一个向量（即索引），该向量对文档的结构和文本信息进行编码，3）使用ASPE加密向量。

实验中的每个数据集都有一个DTD。 我们通过在三个数据集的三个DTD上添加根节点来构建域层次结构（第3.1节）。 因此，每个DTD成为域层次结构的根节点下的子树。Inproceedings数据集中的出版物通常包含多个作者，作者的数量因出版物而异。 我们扫描了整个数据集，发现大多数出版物最多有10位作者。 因此，在表示Inproceedings的DTD的子树中，我们将作者节点复制10次。 在此扩展之后，此子树包含91个节点。 另外两个子树的大小，分别代表另外两个数据集（即股票行情和大学课程）的DTD，分别为19和15（表1）。 因此，域层次结构包含91 + 19 + 15 = 125个节点（不包括根）。 我们通过Dewey标签标记域层次结构，并排序Dewey标签。

一旦域层次结构准备就绪，我们就开始将每个XML文档编码为一个向量。 我们首先将每个XML文档嵌入到域层次结构中。 **特别是，我们使用[1]中提出的算法。将XML文档分解为边缘。**对于每个这样的边缘，算法在域层次结构中找到其潜在的匹配边缘。 所有这些潜在的匹配边缘连接在一起。 如果其节点之间的PC / AD关系与XML文档中的PC / AD关系一致，则连接结果是有效嵌入。 在XML文档的所有可能嵌入中，我们选择最左边的一个（第5节）。 然后，根据Dewey标签及其在域层次结构中的排序，为XML文档构建125维向量。 表2显示了每个数据集的编码时间。



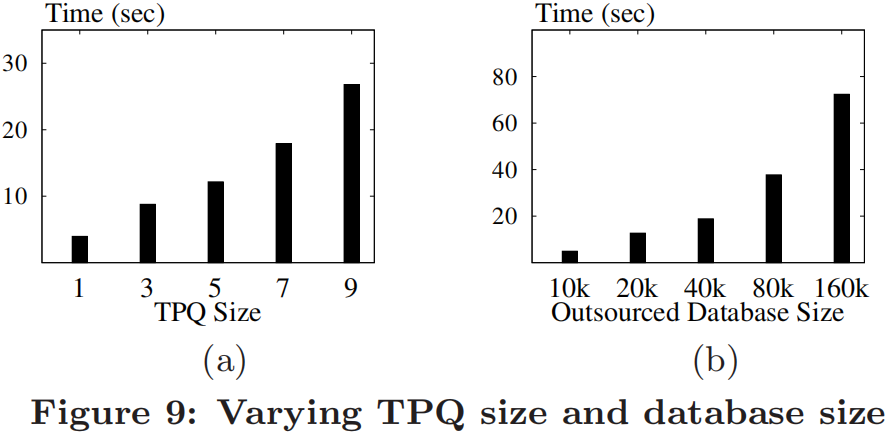
**我们使用ASPE [24]（第3.2节）来加密XML文档的向量（即索引）。**表2给出了每个数据集加密的时间。如果我们一起分析表1和表2，我们可以看到加密时间随着数据集大小的变化呈线性增长。这与分析（在3.3节中）一致，该分析表明加密矢量的时间复杂度为Θ（| U | 2）。 在这里，| U | = 125。

7.2未优化的查询评估

在没有任何优化的基本方案中，所有三个数据集的安全索引被放在一起。 给定任何TPQ，应根据每个加密索引对其进行评估。

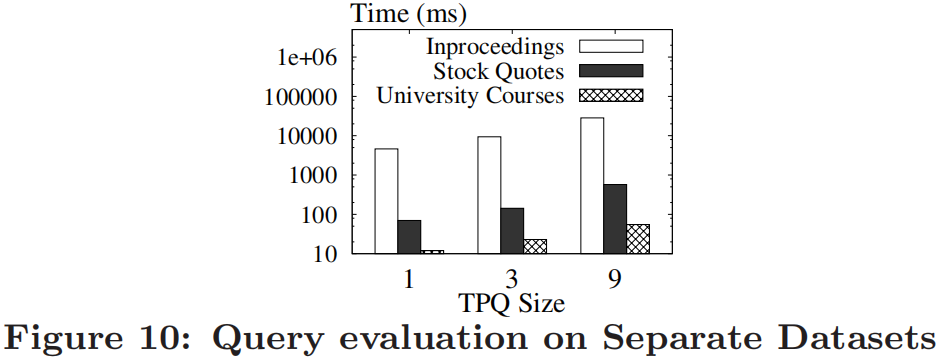
我们首先分析了TPQ大小对查询效率的影响。 我们生成5个TPQ，分别包含1,3,5,7和9个节点。 然后根据第3.3节中的查询构造步骤创建对这5个TPQ的查询。 图9（a）显示了结果，其中经过的查询处理时间作为TPQ大小的函数线性增长。 发生这种情况，因为必须为TPQ中的每个节点创建子查询。

接下来，我们将TPQ大小设置为3，并检查外包数据库大小对查询效率的影响。 我们实验中完整的外包数据库包括三个数据集 - Inproceedings，Stock Quotes和University Courses。 我们通过从完整数据库中随机抽取10k到160k文档来生成5个数据库。 正如预期的那样，查询处理的经过时间随着数据库大小的增加呈线性增长（图9（b））。



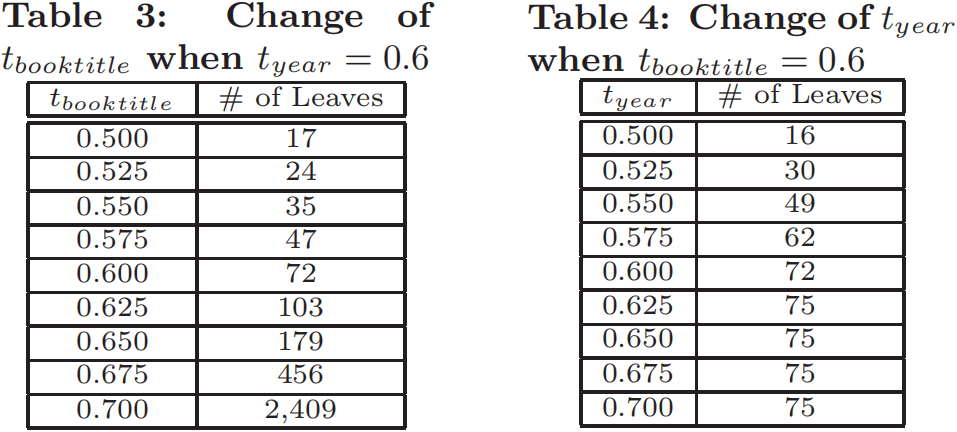
7.2优化过的查询评估

在本节中，我们将介绍应用优化技术后的实验结果。如第4节所述，提高效率的一个简单策略是通过DTD对XML文档进行分区，这样在每个结果分区中，所有XML文档都共享相同的DTD。对于每个分区，域层次结构仅基于其DTD构建，然后根据此域层次结构构建其XML文档的安全索引。在我们的实验中，我们有三个DTD。因此，完整的外包数据库分为三个分区。对于每个分区，我们分别生成3个TPQ，大小分别为1,3和9。图10报告了结果，其中经过的时间随查询大小的增加而增加。此外，较小分区（即股票行情和大学课程）的时间成本小于1秒，远低于较大分区（即，Inproceedings）的时间成本。这是预期的，因为现在只在一个分区中使用XML文档评估TPQ。因此，我们可以看到上述策略对小分区上的TPQ特别有效。

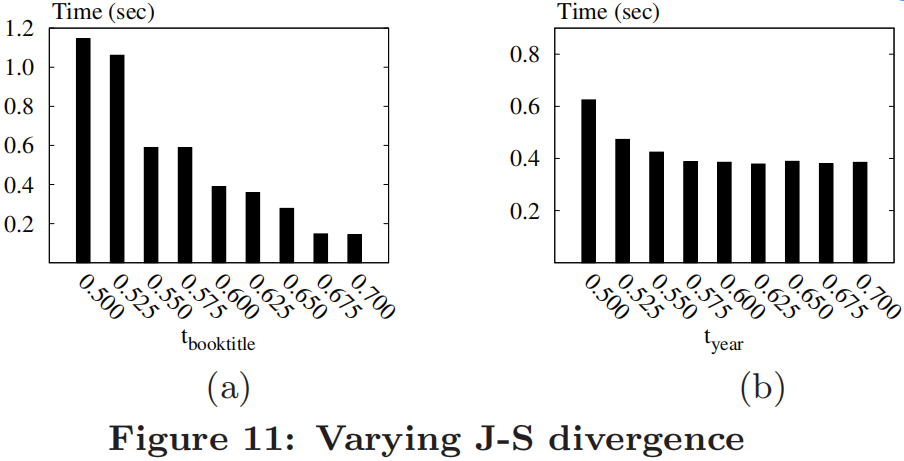


尽管通过DTD对文档进行分区可以在一定程度上提高查询效率，但是仍然存在一些非常大的分区（例如，Inproceedings），并且对它们进行评估是非常耗时的。为了进一步提高效率，我们采用了k-d树，它将Inproceedings分成更小的分区。节点年份和节点书签存在于Inproceedings的XML文档中。年份的价值范围从1959年到2002年，书签有1,744个不同的值。我们将这两个节点作为k-d树的维度，其中k = 2.我们使用定义4中指定的分裂合格条件来确定是否可以进一步分割k-d树中的节点。对于与年份相关联的维度，我们设置阈值tyear，这要求k-d树节点中年份值的局部分布不应与整个Inproceedings数据集中的年份值（通过J-S差异）差异超过tyear。以类似的方式，我们为尺寸booktitle设置阈值tbooktitle。

构建好k-d树后，将仅使用k-d树的叶子中的XML文档来评估具有年份和书签限制的每个点查询。这提高了效率，但是year和booktitle的分布可能因树节点而异。 因此，为了满足分裂合格条件，k-d树中的叶子大小可能不同于一片叶子。 为了更好地衡量评估时间，我们随机生成100个查询，其约束在年份和书签的域中统一生成。



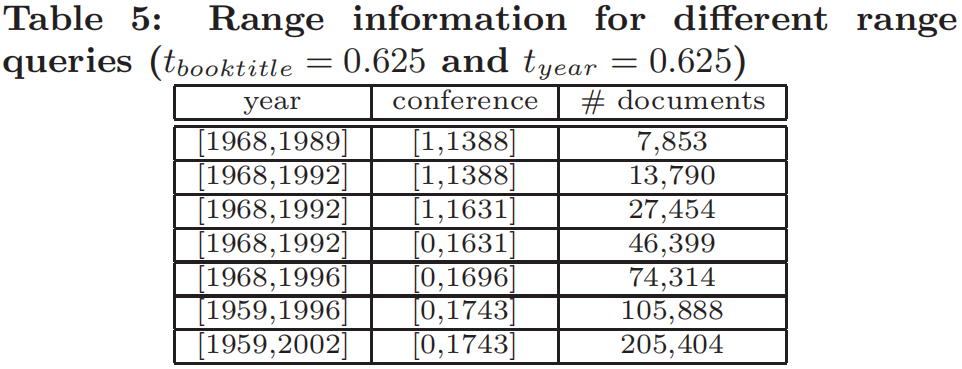
为了看到改进效果，我们首先将年龄的J-S发散阈值固定为0.6，并改变书签的阈值tbooktitle。 表3列出了每个tbooktitle配置的叶子总数。 可以看出，当J-S散度增加时，叶子的总数增加。 这是预期的，因为较大的发散阈值更能容忍书签的全局和局部分布之间的差异，并且可以产生更多的叶子。 随着叶子数量的增加，叶片大小平均减少。 因此，改进了与叶子大小成线性的查询效率。 图11（a）证明这一点。 此外，它还表明，当tbooktitle> 0.625时，效率提高更加明显，因为超过该阈值，叶子的数量以陡峭的方式增加。

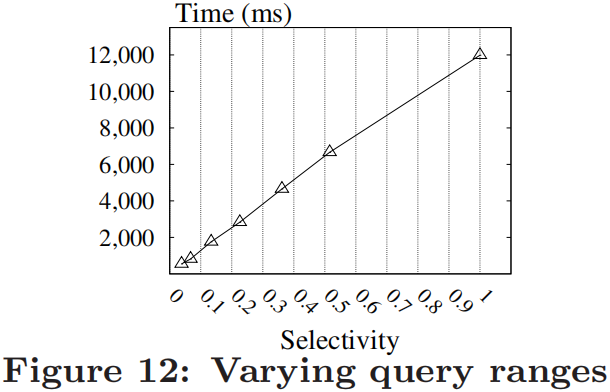


接下来，我们将tbooktitle改为0.6并改变tyear。 对于每个tyear配置，k-d树中产生的叶数在表4中给出。当tyear达到0.625时，产生的叶数停止增加。 这是由于我们对tbooktitle施加的严格约束，这会阻止k-d树节点被进一步分割。 图11（b）中给出的评估时间也反映了这一事实; 它在tyear达到0.625后停止下降。

我们还评估了构建k-d树的时间成本。 k-d树的分裂由J-S阈值引导。较高的J-S阈值放宽了叶节点中数据分布的要求，因此可以构建更大的k-d树并且时间成本更高。但是k-d树的建立仍然是有效的。在所有的实验情况中，它都只需要不到2,676毫秒的时间。

除了点查询之外，我们还考虑了k-d树对范围查询的效率改进的影响。我们针对年份和书签的维度生成7个范围查询。表5中给出了每个范围查询的最小边界框和需要比较的XML文档的数量。图12报告了经过的时间，其中，选择性表示评估给定查询所涉及的文档的比例。 随着选择性的增加，需要为查询评估的XML文档数量也会增加。因此，时间成本也增加。





1. 相关工作

保护关系数据库中的数据隐私已经被广泛研究了[20,11,12,9,4]。 Hacig¨um¨us等人 [11]将每个属性域划分为范围，并按范围将元组分组到桶中。存储桶中的元组是加密的，并且存储桶被分配随机ID。 给定查询，具有可能与查询重叠的元组值的桶将返回给用户。查询结果由用户通过后处理步骤重新定义以消除误报。Damiani等人[9]提出了一种基于直接加密和散列的加密关系数据索引方案。他们进一步分析了索引改进的效率与其披露的额外信息之间的利益关系。

XML文档包含结构和内容信息。因此，对加密的XML文档的安全查询处理更具挑战性。该领域的拟议解决方案包括[4,23,26]。布林克曼等人[4]使用关系表索引XML文档的结构信息，并将表存储在第三方，因此这种方法损害了结构的隐私性。Wang和Lakshmanan [23]根据用户指定的安全约束选择性地将XML文档中的子树分组为块，以隐藏结构信息。但它为每个节点（块）分配一个间隔，以便子节点的间隔包含在其父节点的间隔中，因此一些结构性信息仍然泄露。此外，[23]中的方法假定XML文档中的某些节点不敏感，并且其内容不需要加密。但是，有时候决定哪些数据是敏感的，哪些数据不敏感是很困难的。攻击者甚至可以从部分显示的明文信息中推断出敏感信息[14]。杨等人[26]通过元组对XML文档中的每个根到叶路径进行编码，然后将生成的元组外包给运行关系数据库系统的云服务器。他们的解决方案支持安全的XPath查询但是，由于每个路径都是独立编码的，因此服务器无法处理包含多个相关路径的TPQ。感兴趣的读者可以参考[21]获得有关XML文档外包的更完整的调查。

对加密图形结构数据的查询处理也与我们的工作有关。[6]是一种方案，它将包含查询图的所有图形作为子图返回。它首先挖掘一组频繁的子图（来自整个图数据集）作为特征。 然后，给定查询图，它将提取查询中存在的所有功能。包含查询中的要素的数据集中的所有图形都将作为可能的候选对象返回给用户。最后，用户通过后处理步骤修剪所有误报。这种方法可能适用于搜索XML文档。但是，它仅支持结构匹配，并且缺乏查询节点值的灵活性。 此外，它还可能导致高误报率：对于不包含任何功能的查询，将返回整个数据库。

1. 结论和未来的工作

在本文中，我们提出了一种有效的方法来评估加密XML文档上的TPQ。关键的新颖之处在于我们将XML文档中的信息和TPQ编码为向量。以这种方式，TPQ和文档之间的匹配被减少到计算它们相应的向量之间的距离。我们的方法将确切的查询结果返回给数据用户，这样他们就不需要像现有方法那样使用后处理步骤来修剪错误的候选者。我们还提出了修剪不可匹配的XML文档的优化技术，以提高查询效率。此外，我们的方法支持点和范围查询。广泛的实验结果表明它具有很好的效果，并且可以很好地适应大型数据集。

未来可能的一项工作是在数据用户和云服务器之间部署代理。这样，代理可以减轻数据用户查询编码/加密的负担。到目前为止，我们的工作仅支持字符串值之间的相等匹配。因此，我们研究议程的另一项可能的未来工作是包含模糊字符串匹配的功能。