(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102033956 A
(43)申请公布日 2011.04.27

(21)申请号 201010606470.4

(22)申请日 2010.12.27

(71)申请人 陆嘉恒
地址 100872 北京市海淀区中国人民大学青年公寓 4122 室

(72)发明人 陆嘉恒 林春彬

(51)Int.Cl.
G06F 17/30(2006.01)

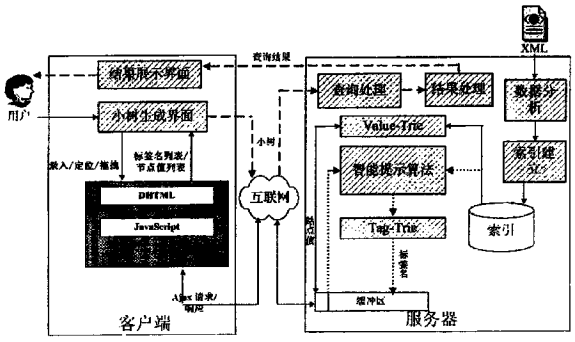
权利要求书 3 页 说明书 8 页 附图 5 页

(54)发明名称

具有智能提示功能的图形化 XML 内容和结构查询系统

(57)摘要

本发明属于信息技术领域关于智能化的 XML 数据查询技术，具体的是在提出新的 XML 图形化查询方法的基础上设计了一套具有智能提示功能的 XML 查询系统。该系统提供在一个图形化创建查询的界面上，针对标签名称和结点值提供了智能提示功能来引导用户创建查询。从而使用户在不了解 XML 模式、内容、结构甚至不会写 XQuery 等查询语言的情况下也能进行 XML 搜索。系统定义了一个综合考虑查询结果的排序机制，并把对每个结果进行打分后用树的形式展示给用户；1，开发支持图形化拖拽方式创建 XML 查询语句的界面；2，设计智能提示算法；3，设计对查询结果根据结构和内容信息排序的机制；4，开发为结果打分并以树的显示展示结果的界面。



CN 102033956 A

1. 具有智能提示功能的图形化 XML 数据的查询技术，具体的步骤包括：

A、开发一个图形化创建小枝模式的操作界面。

B、设计智能提示算法，能根据小枝中结点的不同位置来自动反馈当前该结点可行的标签名称，并设计 tag-trie 来支持前缀搜索。

C、设计能自动识别结点类型并且动态构建 value-trie 来支持对大量结点值的前缀搜索。

D、设计新的既考虑结构又考虑内容的排序公式

E、开发能以树形方式展示结果，并对查询结果进行打分界面。

2. 根据权利请求 1 中的步骤 A，其具体步骤包括如下内容：

步骤 A1：实现拖拽圆（系统中给定的，并且支持多次拖拽）到任意的地方来产生结点，在拖拽后的结点的右下角出现两个输入框：标签名称（必填）和结点值（可选）。

步骤 A2：根据权利申请 1 中 B 和 C 的方法分别完成对这两个输入框的信息提交，结点则创建成功。

步骤 A3：选定两个结点，并且确定要连接的边的类型，然后进行连线。连线后一棵小枝创建成功。

步骤 A4：如果要对某个结点增加顺序要求，则选中那个结点，点击增加顺序按钮。

3. 根据权利请求 2，其中步骤 A1 进一步阐述为：

步骤 A1-1：事先在系统中放置 N 个 ID 不同的圆（c_i，i 表示第 i 个圆），其中 1 个可见 N-1 个不可见，每个圆都有一个变量 flag 来控制它是否可以被拖拽。N 个 ID 不同的不可见的层（l_i，i 表示第 i 个层），每个层中包含四个输入框，两个可见的用于输入标签名（node_name）和结点值（node_value），两个不可见的用于记录提交信息时对应结点的 X，Y 坐标（xpos 和 ypos）；还需放置三个不可见的输入框，一个用于记录当前的小枝情况（twig_current），另一个用于记录哪个结点被标志了顺序（node_order），最后一个记录所有结点信息串（node_message），格式为［结点名，结点值，X 坐标，Y 坐标］；

步骤 A1-2：利用 JavaScript 实现当对第一个圆进行拖拽时，原来不可见的第二个圆变得可见，以此类推；

步骤 A1-3：利用 JavaScript 实现当第一个圆被拖拽到一个地方后，第一个层变得可见，并出现在该结点的右下角，并把该结点的 X，Y 坐标记录到 xpos 和 ypos 中。

4. 根据权利请求 2，其中步骤 A2 进一步过程为：

步骤 A2-1：当用户把鼠标定位到标签名称的输入框时，JavaScript 调用 AJAX 发送请求，把当前结点的 xpos 和 ypos 发送到服务器，然后根据权利申请 1 中的步骤 B 在输入框旁边出现了一个存放目前可行的标签名称列表。

步骤 A2-2：用户选择列表中的任意一个，则 node_name 输入框中的值也跟着变化。如果用户通过键盘直接在 node_name 输入框中输入字符，则用户每输入一个字符就触发一次 AJAX 请求的发送，则根据权利申请 1 中的步骤 B 返回一个前缀与用户输入信息相同的，并且在目前位置下是可行的标签名称列表，并更新 node_message 的值。

步骤 A2-3：当用户选定一个结点名称后并把鼠标定位到 node_value 输入框中时，触发一次 AJAX 请求，则根据权利申请 1 中的步骤 C 反馈该标签名称下的所有值中具有代表性的几个，当用户在 node_value 中输入字符时，用户每输入一个字符就触发一次 AJAX 请求的发

送，由于值的个数一般都很多，所以根据权利申请 1 中的步骤 C 返回一些前缀与用户输入信息相同的并具有代表性的值，并更新 node_message 的值。

5. 根据权利请求 2，其中步骤 A3 进一步过程为：

步骤 A3-1：利用 JavaScript 实现平面上任意两点间的连线，定义 drawPC(x1, y1, x2, y2) 方法来对点 (x1, y1) 和 (x2, y2) 连接一条红线，表示 Parent-Child 关系；drawAD(x1, y1, x2, y2) 方法来对点 (x1, y1) 和 (x2, y2) 连接两条平行的绿线，表示 Ancestor-Descendant 关系。

步骤 A3-2：利用 JavaScript 从 node_message 中获取已经创建好的结点名称和当选定要连线的两个结点对应的 X，Y 坐标。

步骤 A3-3：当选择要连接 Parent-Child 边时，系统会调用 drawPC 方法来加边。当选择 Ancestor-Descendant 时，则调用 drawAD 方法来加边。每次为两个结点加边的操作实际上就是创建小枝 (Twig) 的操作，因此要更新 twig_current 的值。

6. 根据权利请求 2，其中步骤 A4 进一步过程为：

步骤 A4-1：利用 JavaScript 实现 markOrder(x, y) 方法在任意一点增加顺序标记。

步骤 A4-2：利用 JavaScript 从 node_message 中获取要增加顺序要求的结点对应的 X，Y 坐标，然后调用 markOrder 来增加“<”标记到相应的结点上。

步骤 A4-3：设置 node_order 的值为被被标记了顺序的结点，并更新 twig_current 的值。

7. 根据权利请求 1，其中步骤 B 进一步过程为：

步骤 B1：当用户把鼠标定位到 node_name 输入框中时，AJAX 会把 xpos 和 ypos 中的值发给服务器。

步骤 B2：服务器根据传来的值调用智能提示算法，把所有可行的结果反馈出来，并构建 tag-trie 来支持用户的前缀搜索。

8. 根据权利请求 1 中的步骤 C，其具体步骤包括如下内容：

步骤 C1：当用户把鼠标定位到 node_value 输入框中时，AJAX 会把 node_name 中的值发给服务器。

步骤 C2：服务器根据传来的节点名称在预处理 XML 文档时记录下的统计信息中找该结点下的值的类型。如果为数值型，则返回一个具有描述大于，小于，等于以及“and”与“or”操作的下拉框组；如果为字符串型则返回一个输入框即可。

步骤 C3：当判断出结点值的类型为字符串时，就要选择结点值中具有代表性的一些进行构建 value-trie 来支持前缀搜索。

步骤 C4：当用户在 node-value 中输入字符时，之前的 value-trie 返回的结果个数如果少于 5 个或者不存在以这些字符为前缀的值时，会删除该 value-trie，继续根据索引寻找符合前缀要求的值，并选择这些值中的少数有代表性的，再构建新的 value-trie 来提供前缀搜索。直到用户在列表中找到自己想要的值为止。

9. 根据权利请求 1，其中步骤 D 的进一步过程为：

步骤 D1：提出公式 S_V 来对结果按照内容进行排序

步骤 D2：提出公式 S_T 来对结果按照结构进行排序

步骤 D3：提出公式 Score 综合考虑 S_V 和 S_T 来对结果进行排序。

10. 根据权利请求 9，其中步骤 D1 的进一步过程为：

$$S_V(r)=\sum_{c\in r} wf-idf(c,r)$$

wf-idf(c，r) ＝ wf(c，r)*idf(c)

$$wf(c,r)=\begin{cases}1+\log(tf(c,r)) & if\ tf>0\\ 0 & otherwise\end{cases}$$

其中，c 是各个结果 r 中一个术语。

11. 根据权利请求 9，其中步骤 D2 的进一步过程为：

$$S_T(r,q)=\sum_{(pq,pr)\in P}\left\{\left(\frac{1+|pq|}{1+|pr|}\right)*\sum_{t_{pq}\in pq}\frac{wf\text{-}idf(t_{pq},r)}{wf\text{-}idf(t_{pq},q)}\right\}-\frac{\sum_{t_T\in T} wf\text{-}idf(t_T,r)}{\sqrt{\sum_{t_r\in r} wf\text{-}idf(t_r,r)^2}}$$

其中 {pq，pr} 表示从查询 q 和结果 r 中的路径的匹配。T 表示那些出现在结果 r 中但是不出现在查询 q 中的结点类型的集合。参数 t_{pq}，t_r t_T 分别表示来自路径 pq，结果 r 和集合 T 的结点类型。

12. 根据权利请求 9，其中步骤 D3 的进一步过程为：

score(r，q) ＝ α S_V(r)+(1-α)S_T(r，q)

其中 α 是一个在介于 0，1 之间的系数，而且 S_V 和 S_T 分别表示值和结构因素。

13. 根据权利请求 1，其中步骤 E 的进一步过程为：

步骤 E1：查询结果返回的是 XML 小文档，用 jsp 来计算文档中结点，结点值，属性等赋予能构成树形式的坐标（即根结点的 y 坐标最小，叶子结点的 y 坐标最大，x 坐标的分配按照各层的结点个数多少来动态改变）。

步骤 E2：用 JavaScript 在这些坐标点上绘制相应的圆（表示结点），方形（表示结点值）或者菱形（表示属性）。

步骤 E3：用 JavaScript 实现类似权利申请 5 步骤 A3-1 中的 drawPC 那样的画线功能，这里要实现三种边：结点与结点，结点与值，结点与属性。

步骤 E4：根据步骤 E1 的求得的坐标，以及识别每个坐标对应的元素来来相应的线条。

步骤 E5：根据权利申请 8 步骤 D3 中求出的 score 的值，为每个结果绘制出一个以该值为宽度的有背景色的层，并绘制一个宽度为 100 的，无背景色但是有边框颜色的层放在后面。这样就能直观地表示出各个结果的分数。

具有智能提示功能的图形化 XML 内容和结构查询系统

技术领域

[0001] 本发明涉及信息检索或数据查询等应用领域中 XML 查询的应用，尤其是当用户不了解要查询的 XML 文档的内容，结构，模式等信息的情况下的图形化 XML 小枝查询技术。

背景技术

[0002] 已有的 XML 数据库允许用户提交一个 XPath 或者 XQuery 来查询相关的结果。这种信息查询模式需要用户对 XML 数据库的结构和内容比较了解，而且要对查询语言很熟悉。如果一个用户对 XML 了解很少，则在写查询时他经常会感到困惑，所以他不得不去学习复杂的 DTD 或者 XML 模式从而来了解 XML 文档的结构。

[0003] 为了解决这些问题，学者们已经设计了多种系统旨在简化用户的查询过程。已有的图形化 XML 查询系统大致可以分为两类 :(i) 开发图形化系统支持传统的 XQuery 语言，例如 XQE，这些系统提供友好的接口来输入 XQuery 表达式并且作了大量的工作来优化查询计划。(ii) 开发图形化查询语言来把查询转化为 SQL 或者 SQL/XML，例如 GLASS，XML-GL，这些系统提供新的图形化语言来查询 XML 文档。但是上述这些系统依然要求用户对 XML 文档的结构和内容的知识，但是这对于普通用户而言是很困难的，而且随着 XML 文档越来越复杂，越来越大，要在查询前事先对文档进行了解是很不合理，很不科学的。而且不利于推广。

发明内容

[0004] 为了克服已有 XML 图形化查询系统要求用户必须对 XML 文档的结构或者内容了解的不足，本发明开发新颖的图形化界面来提供 XML 小枝查询下的智能提示功能，以此来智能引导用户完成查询条件的创建，并用新的排序公式来对结果进行排序，最后用图形化界面来直观展示结果。

[0005] 本发明首先利用 JavaScript+jsp 设计了支持图形化方式创建小枝模式的系统。该步骤 A 的主要实现步骤包括 :A1、实现通过拖拽产生结点。A2、根据 B 和 C 返回的列表来确定结点。A3、通过为结点加边来完成小枝模式的创建。A4、通过增加顺序标志来实现带顺序的查询。

[0006] 其次，本发明提出了智能提示算法来返回标签名称列表。该步骤 B 主要实现步骤包括 :B1、把正在创建的结点的 X，Y 坐标通过 AJAX 发送给服务器。B2、通过智能提示算法返回目前可行的标签名称。B3、为这些标签名称构建 tag-trie。

[0007] 接着，本发明提出了结点值提示方案。该步骤 C 主要包括 :C1、通过 AJAX 把 node_name 中的值发给服务器。C2、通过预处理时的统计信息来来判断该结点下的值的类型 :为数值型还是字符串型，并根据不同类型返回相应的输入框。C3、根据索引找到属于该标签名的值，并抽取具有代表性的几个来构建 value-trie。C4、通过 AJAX 把用户在 node-value 中输入字符发给服务器，服务器会调整抽取的值来构建新的 value-trie。

[0008] 进一步，本发明设计了新的结果排序公式。该步骤 D 主要包括 :D1、设计考虑内容

的排序公式 S_V。D2、设计考虑结构的结果排序公式 S_T。D3、设计公式 Score 把 S_V 和 S_T 结合起来对结果进行排序。

[0009] 最后，本发明利用 JavaScript+jsp 设计了结果展示和打分系统。该步骤 E 包括：E1、求得结果中各个元素的坐标。E2、用 JavaScript 在 这些坐标点上相应的图形来表示相应的元素。E3、用 JavaScript 为各个元素连接相应的边。E4、用 JavaScript 绘制相应长度的进度条来表示公式 score 的值。

[0010] 本发明的有益效果是，提供高效的智能提示功能来改进传统的图形化 XML 搜索系统必须要求用户事先对要搜索的 XML 文档的结构和内容了解的不足，使得用户（即使是对文档一无所知的初级用户）能方便地创建有意义的小枝查询。而且本发明设计了新颖的图形化创建小枝模式的界面。本发明还提出了新的排序公式综合考虑了内容和结构信息，很好地对结果进行排序，并能通过图形化的方式直观地展示搜索结果给用户。

附图说明

[0011] 图 1：XML 文档转化为一棵有序树。

[0012] 图 2：一个小枝模式在 XML 文档中执行的结果。

[0013] 图 3：由 {b，abc，abd，bcd，abcd，efg，hii} 这六个词构成的 Trie 树的结构。

[0014] 图 4：有无顺序限制时智能提示的不同。

[0015] 图 5：查询带顺序要求的航班信息时的小枝模式。

[0016] 图 6：系统结构图

[0017] 图 7：是否在一个结点的影响范围内对潜在结点名称的影响

[0018] 图 8：新结点落在多个结点的范围内时可行结点名称必须保证是其他所有结点的后代。

[0019] 图 9：父亲结点是否有顺序要求时可行结点名列表的不同。

[0020] 图 10：能自动考虑结点在小枝中的位置以及结点间的关系并为结点返回智能提示的算法。

具体实施方式

[0021] 为了更全面地理解本发明及其优点，下面结合附图及具体实施过程对本发明做进一步详细地说明。

[0022] 一．为了便于清楚理解期间，首先对以下几个定义进行简单地介绍。

[0023] XML 文档：XML 文档由各种用户自定义的标签有结构有顺序地组成，可以转化为一棵带根结点的有序树。图 1 是 XML 文档转化为有序树的例子。

[0024] XQuery 语言：是另外一种 XML 查询语言，XQuery 相对于 XML 的关系，等同于 SQL 相对于数据库表的关系。XQuery 使用函数来提取 XML 文档中的数据。XQuery 还定义了 FLWOR 结构来进行规范查询，其中：FLWOR 是" For，Let，Where，Order by，Return" 的首字母缩写。下面是一个 XQuery 表达式：

[0025]

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
return $x/title
```

[0026] 这个查询表达式是要选取 bookstore 元素下的 book 元素下所有的 title 元素，并且其中的 price 元素的值必须大于 30。

[0027] 小枝模式 (twig pattern)：一个 XML 小枝模式就是一棵节点表示标签，属性或者文本，边表示父子关系（一般用一条边表示）或者祖孙关系（一般用两条平行边表示）的小枝。Twig Pattern 常常在 XML 树枝查询时使用。小枝模式查询是 XQuery 的核心操作，执行 XQuery 时实际上是先转化为相应的小枝模式。

[0028] 小枝模式查询：在给定的一个 XML 数据集 D 中，把内容和结构都 满足小枝模式 Q 的结果求出来。图 2 展示了一个小枝查询在 XML 文档中查询得到的结果。该例子中的小枝模式的含义是“t 与 f 为 s 的后代”，只要满足这个小枝模式的都是结果。因此结果有：<s1, t1, f1>, <s2, t2, f1>, <s1, t2, f1>。

[0029] Trie：也称为字典树，利用字符串的公共前缀来节约存储空间。Trie 树每一层的节点数是 26^i 级别的。Trie 树有三点性质：1. 根结点不包含字符，除根结点外每一个结点都只包含一个字符；2. 从根结点到某一结点，路径上经过的字符连接起来，为该结点对应的字符串；3. 每个结点的所有子结点包含的字符都不相同。图 3 是一个由字符串 {b, abc, abd, bcd, abcd, efg, hii} 这 6 个单词构建成的 trie 树。当给定一个关键字时，trie 树进行如下步骤的搜索：(1) 从根结点开始一次搜索；(2) 取得要查找关键词的第一个字母，并根据该字母选择对应的子树并转到该子树继续进行检索；(3) 在相应的子树上，取得要查找关键词的第二个字母，并进一步选择对应的子树进行检索；(4) 迭代过程……(5) 在某个结点处，关键词的所有字母已被取出，则读取附在该结点上的信息，即完成查找。查找的时间复杂度为 O(n)，n 为字符串长度。例如有关键字“ab”，则根据上面的搜索步骤，返回的结果为 abc, abcd, abd。这就为前缀搜索提供了很好的方案。

[0030] Tag-trie：由上面的 trie 介绍可知，trie 树消耗的内存是比较大的，但是搜索效率是很高的，特别在前缀搜索中。因为 XML 文档中的标签名称基本都是字母，所以 tag-trie 每一层的节点数最多是 26^i 级别的。而且标签名称的个数往往都不多，例如 dblp(130M) 的公开数据集中的标签名也才不过 30 多个。因此考虑到 XML 标签名的组成元素单一，个数少的特性，我们就采用标准的 trie 树方法来构建 tag-trie 来对标签名称进行搜索。

[0031] Value-trie：XML 中结点值往往个数是很多的，而且组成值的元素往往包含数字，字母，符号等。如果直接用标准的 trie 为这些值构建 value-trie，那么内存很容易崩溃。这样大的消耗是很不科学的。但是在智能提示中输入结点值时又必须提供前缀搜索的功能。于是本发明中采用了动态构建 value-trie 的方法。即事先为结点值建立索引，把属于同一个标签名下的值放到一个物理块中，并把每个物理块中的值按照字典顺序进行排序，并把第一个字符相同的值归到一个小块中，再把小块中的前两个字符相同的归到更小块中……以此类推，把值按照前缀的相似度在物理块上划分开来。

[0032] 当用户输入字符时，利用索引快速定位到以该字符为前缀的值所在的地址，并从

中抽取出一些具有代表性的值来为这些值动态构建 value-trie 树，这样就保证了值的个数少却不缺乏代表性，也保证了 value-trie 的大小和效率。

[0033] 顺序：在 XML 搜索中，带顺序的搜索在 XPath 和 XQuery 中已经运用得比较多了（例如 XPath 中的 following-sibling 和 preceding-sibling 等），但是在小枝模式查询中还没有得到很好的表示和处理。顺序要求也就是要求标签在 XML 文档中出现的顺序。

[0034] 图 4 很好地说明了没有顺序要求和有顺序要求时查询结果的不同。在图 4(a) 中当没有顺序要求时，这个小枝模式查询的结果为两个：(s1, t1, f1) 和 (s1, t2, f1)。

[0035] 在图 4(b) 中当在这个小指模式上增加顺序要求时（这里用一个红色的”<”表示 s 结点的孩子必须有顺序）。则这个小枝模式查询的结果就剩一个 (s1,t1,f1) 了，而 (s1, t2, f1) 不再是结果，因为在文档中的 t2 是在 f1 之后的，而小枝模式中要求 t 在 f 之前。

[0036] 传统的小枝模式是不能表示顺序的，而为传统的小枝模式拓展了表示顺序功能后，用户可以表示更丰富的查询，例如用户可以查找“起飞时间为 10 点的航班的降落时间，并且起始时间应在降落时间之前。这用传统的小枝模式是没办法表示的，而我们用增加了顺序的小枝模式则可以很轻松地表示，图 5 就是该查询对应的一个小枝模式。

[0037] 二、以下内容详细说明本发明的具体实施步骤：

[0038] 步骤 A，开发图形化创建小枝模式系统。

[0039] 下面结合图 6、图 7 说明步骤 A 的具体实施步骤：

[0040] 图 6 中的“小树生成界面”部分对应的也就是步骤 A 的内容。

[0041] 步骤 A1：开发一个面板，用于实现通过拖拽圆来产生结点，在拖拽后的结点的右下角出现两个输入框：标签名称（必填）和结点值（可选）。

[0042] 步骤 A1 中的过程可进一步分解为以下几个步骤：

[0043] 步骤 A1-1：事先在系统中放置 N 个 ID 不同的圆 (c_i，i 表示第 i 个圆)，其中 1 个可见 N-1 个不可见，每个圆都有一个变量 flag 来控制它是否可以被拖拽。N 个 ID 不同的不可见的层 (l_i，i 表示第 i 个层)，每个层中包含四个输入框，两个可见的用于输入标签名 (node_name) 和结点值 (node_value)，两个不可见的用于记录提交信息时对应结点的 X，Y 坐标 (xpos 和 ypos)；还需放置三个不可见的输入框，一个用于记录当前的小枝情况 (twig_current)，另一个用于记录哪个结点被标志了顺序 (node_order)，最后一个记录所有结点信息串 (node_message)，格式为 [结点名，结点值，X 坐标，Y 坐标]；

[0044] 步骤 A1-2：利用 JavaScript 实现当对第一个圆进行拖拽时，第一个圆跟随鼠标移动到任意位置，同时原来不可见的第二个圆变得可见，以此类推；

[0045] 步骤 A1-3：利用 JavaScript 实现当第一个圆被拖拽到一个地方后， 第一个层变得可见，并出现在该结点的右下角，并把该结点的 X，Y 坐标记录到 xpos 和 ypos 中。

[0046] 步骤 A2：通过为两个输入框输入值来确定正在创建的结点的信息。

[0047] 步骤 A2 中的过程可进一步分解为以下几个步骤：

[0048] 步骤 A2-1：当用户把鼠标定位到标签名称的输入框时，JavaScript 调用 AJAX 发送请求，把当前结点的 xpos 和 ypos 发送到服务器，然后根据权利申请 1 中的步骤 B 在输入框旁边出现了一个存放目前可行的标签名称列表，图 8(a) 就是服务器为一个结点返回的列表的形式。

[0049] 步骤 A2-2：用户选择列表中的任意一个，则 node_name 输入框中的值也跟着变化。

如果用户通过键盘直接在 node_name 输入框中输入字符，则用户每输入一个字符就触发一次 AJAX 请求的发送，则根据权利申请 1 中的步骤 B 返回一个前缀与用户输入信息相同的，并且在目前位置下是可行的标签名称列表，并更新 node_message 的值。

[0050] 步骤 A2-3：当用户选定一个结点名称后并把鼠标定位到 node_value 输入框中时，触发一次 AJAX 请求，则根据权利申请 1 中的步骤 C 反馈该标签名称下的所有值中具有代表性的几个，当用户在 node_value 中输入字符时，用户每输入一个字符就触发一次 AJAX 请求的发送，由于值的个数一般都很多，所以根据权利申请 1 中的步骤 C 返回一些前缀与用户输入信息相同的并具有代表性的值（图 8(b) 就是服务器返回的在 author 标签下，开头为 Ah 的部分值的列表），并更新 node_message 的值。

[0051] 步骤 A3：通过为结点加边来生成小枝模式。

[0052] 步骤 A3 中的过程可进一步分解为以下几个步骤：

[0053] 步骤 A3-1：利用 JavaScript 实现平面上任意两点间的连线，定义 drawPC(x1, y1, x2, y2) 方法来对点 (x1, y1) 和 (x2, y2) 连接一条直线表示 Parent-Child 关系；drawAD(x1, y1, x2, y2) 方法来对点 (x1, y1) 和 (x2, y2) 连接 两条平行的直线表示 Ancestor-Descendant 关系（用一条直线表示父子关系，两条平行的直线表示祖先后代关系是在 XML 研究领域中对小枝模式描述时的规范表示）。

[0054] 步骤 A3-2：利用 JavaScript 从 node_message 中获取已经创建好的结点名称和当选定要连线的两个结点对应的 X，Y 坐标。

[0055] 步骤 A3-3：当选择要连接 Parent-Child 边时，系统会调用 drawPC 方法来加边。当选择 Ancestor-Descendant 时，则调用 drawAD 方法来加边。每次为两个结点加边的操作实际上就是创建小枝 (Twig) 的操作，因此要更新 twig_current 的值。

[0056] 步骤 A4：为小枝模式增加顺序限制。

[0057] 步骤 A4 中的过程可进一步分解为以下几个步骤：

[0058] 步骤 A4-1：利用 JavaScript 实现 markOrder(x, y) 方法在任意一点增加顺序标记。

[0059] 步骤 A4-2：利用 JavaScript 从 node_message 中获取要增加顺序要求的结点对应的 X，Y 坐标，然后调用 markOrder 来增加“<”标记到相应的结点上（图 5 中结点“航班”上面的那个红色标志”<”就表示航班的所有子结点必须是有序的，即查询结果中“起飞时间”必须在“降落时间”之前）。

[0060] 步骤 A4-3：设置 node_order 的值为被被标记了顺序的结点，并更新 twig_current 的值。

[0061] 步骤 B：设计智能提示算法来返回不同位置下的可行的标签名称，并设计 tag-trie 来支持前缀搜索。

[0062] 下面结合图 6、图 8、图 9、图 10 来说明步骤 B 的具体实施步骤：

[0063] 步骤 B1：当用户把鼠标定位到 node_name 输入框中时，AJAX 会把 xpos 和 ypos 中的值发给服务器，图 6 展示了如何从 client 端通过 javascript 发送 AJAX 请求到 server 端。

[0064] 步骤 B2：服务器根据传来的值调用智能提示算法，把所有可行的结果反馈出来，并构建 tag-trie 来支持用户的前缀搜索。图 6 展示了如何从 server 端把 tag-name 通过

AJAX 响应返回到 client 端。

[0065] 首先，规定创建小枝查询时每个结点下方的 150° 为该结点的领域 (scope)。假设有 A，B 两个结点，当 B 结点落在 A 结点的领域内，则 B 就应该是 A 的后代，即 B 的潜在标签名称的集合就是所有 A 的后代的集合，这里用 Desc(A) 表示。图 8(a) 中 X 与 school 构成了 43° 的角度 (小于 75°) 即 X 落在了 school 的领域内。因此 (a) 中的返回列表中的标签名称都是 school 的后代，而 (b) 中 X 没有落在 school 内，没有必须是 school 后代这个限制，因为它与 school 构成的角度为 92° 大于 75° ，所以 (b) 中列表中的标签名比 (a) 中多。

[0066] 而在图 9 中，X 落在了 dblp，school 和 inproceedings 的领域内。特别注意，在这个小枝模式中 dblp 比 school 和 inproceedings 所处的层次高，而 school 和 inproceedings 是同一层次。因此 X 的潜在标签名称为 Desc(dblp) ∩ (Desc(school) ∪ Desc(inproceedings))。这样才能满足实际的需求。

[0067] 因此，在创建小枝模式时，我们采用公式 ：$C=\bigcap_i\left\{\bigcup_j Desc(n_{ij})\right\}$ 来判断一个标签名是否能被加到列表中，这里 n_{ij} 表示小枝查询中第 i 层第 j 个结点。

[0068] 如果新的结点 (称为 newNode) 在某个被标注顺序的结点的范围内时，智能提示算法会识别在小枝查询中 newNode 的左右两边的结点各是哪些 (令左兄弟为 left-sibling 右兄弟为 right-sibling)，然后算法只返回那些在文档中的标签顺序符合 newNode 在 left-sibling 右边且在 right-sibling 左边的结点，图 10(a) 中的 book 没有被标注顺序而 (b) 中被标注了顺序，可以很清楚地看出在 (b) 中的潜在标签比 (a) 少得多，因为 (b) 中的标签在文档中的位置应在 author 的右边，section 的左边。

[0069] 当服务器接收到传过来的 X，Y 参数时智能提示算法首先判断该结点是不是创建的第一个结点，如果是，服务器将把所有的标签名称返回给用户 ；否则，服务器将检查现在界面上的结点间的结构关系和顺序限制，并且用上面的公式来计算潜在的标签名称。如果有顺序限制，那么标签名称就应该满足相应的顺序限制。

[0070] 步骤 C ：设计实时动态构建 value-trie 的方案来支持对大量值的前缀搜索。

[0071] 下面结合图 6、图 7 说明步骤 C 的具体实施步骤 ：

[0072] 步骤 C1 ：当用户把鼠标定位到 node_value 输入框中时，AJAX 会把 node_name 中的值发给服务器，图 6 中显示了值从客户端通过 AJAX 发送到服务器端的过程。

[0073] 步骤 C2 ：服务器获得通过 Ajax 传来的结点名称后，根据预处理时存储的统计信息，判断出该结点下的值的类型 ：数值型或者字符串型。然后根据不同的类型返回不同的输入框给用户输入。图 7(a) 显示的是结点值为数值型时系统返回的输入框的形式，图 7(b) 显示的是结点值为字符串时系统返回的输入框的形式。

[0074] 步骤 C3 ：服务器根据传来的值在利用索引定位到该标签名称下的值所在的位置，由于值特别多，先选取少数代表性的值构建 value-trie 来支持前缀搜索。

[0075] 步骤 C4 ：当用户在 node-value 中输入字符时，之前的 value-trie 返回的结果个数如果少于 5 个或者不存在以这些字符为前缀的值时，会删除该 value-trie，继续根据索引寻找符合前缀要求的值，并选择这些值中的少数有代表性的，再构建新的 value-trie 来提供前缀搜索。直到用户在列表中找到自己想要的值为止，图 6 也展示了 value-trie 是如

何通过 AJAX 响应把查询结果反馈给用户的流程，图 7(b) 展示了反馈回的列表的样式。

[0076] 步骤 D：设计新的既考虑结构又考虑内容的排序公式。步骤 D 的具体设计实现过程可以分为以下几步：

[0077] 步骤 D1：提出公式 S_V 来对结果按照内容进行排序

[0078] $$S_V(r)=\sum_{c\in r} wf-idf(c,r)$$

[0079] wf-idf(c,r) = wf(c,r)*idf(c)

[0080] $$wf(c,r)=\begin{cases}1+\log(tf(c,r)) & if\ tf>0\\ 0 & otherwise\end{cases}$$

[0081] 其中，c 是各个结果 r 中一个术语。

[0082] 步骤 D2：提出公式 S_T 来对结果按照结构进行排序

[0083] $$S_T(r,q)=\sum_{(pq,pr)\in P}\left\{\left(\frac{1+|pq|}{1+|pr|}\right)*\sum_{t_{pq}\in pq}\frac{wf\text{-}idf(t_{pq},r)}{wf\text{-}idf(t_{pq},q)}\right\}-\frac{\sum_{t_T\in T} wf\text{-}idf(t_T,r)}{\sqrt{\sum_{t_r\in r} wf\text{-}idf(t_r,r)^2}}$$

[0084] 其中 {pq，pr} 表示从查询 q 和结果 r 中的路径的匹配。T 表示那些出现在结果 r 中但是不出现在查询 q 中的结点类型的集合。参数 t_{pq}，t_r t_T 分别表示来自路径 pq，结果 r 和集合 T 的结点类型。

[0085] 步骤 D3：提出公式 Score 综合考虑 S_V 和 S_T 来对结果进行排序。score(r，q) = $\alpha S_V(r)+(1-\alpha)S_T(r,q)$ 其中 α 是一个在介于 0，1 之间的系数，而且 S_V 和 S_T 分别表示值和结构因素。

[0086] 步骤 E：开发展示查询结果和为结果打分的系统。

[0087] 步骤 E1：查询结果返回的是 XML 小文档，用 jsp 来计算文档中结点，结点值，属性等赋予能构成树形式的坐标（即根结点的 y 坐标最小，叶子结点的 y 坐标最大，x 坐标的分配按照各层的结点个数多少来动态改变）。

[0088] 步骤 E2：用 JavaScript 在这些坐标点上绘制相应的图形来表示结果中的各个元素，按照国内外 XML 相关论文的习惯，我们采用圆表示结点，方形表示结点值而菱形则表示属性。

[0089] 步骤 E3：用 JavaScript 实现类似权利申请 5 步骤 A3-1 中的 drawPC 那样的画线功能，这里要实现三种边：结点与结点，结点与值，结点与属性。

[0090] 步骤 E4：根据步骤 E1 的求得的坐标，以及识别每个坐标对应的元素来来相应的线条。

[0091] 步骤 E5：根据权利申请 8 步骤 D3 中求出的 score 的值，为每个结果绘制出一个以该值为宽度的有背景色的层，并绘制一个宽度为 100 的，无背景色但是有边框颜色的层放在后面。这样就能直观地表示出各个结果的分数。

[0092] 三．测试结果

[0093] 1. 准备工作：

[0094] 在实验系统中测试了图形化小枝模式查询中智能提示的准确性和高效性。系统中执行查询部分由 JSP1.2 和 Servlet2.5 开发，而创建小枝模式和结果展示部分由 JSP 和

JavaScript 实现。系统运行在 UbuntuLinux 9.10 操作系统中的 Apache Tomcat 6.0 上。实验测试所用的数据集为 130MB 的 dblp 数据集，50M 的 airline 数据集，以及 30M 的 Sigmod 数据集。测试系统已经发布到互联网上，可以通过网址访问：http://datasearch.ruc.edu.cn:8080/LotusX/

[0095] 2、提示的重要性

[0096] 实验中使用 130MB 的 dblp 数据集。在不知道各个数据集的结构、内容时，用户不知道该搜索什么，更不知道该怎么搜索，留给用户的只有疑惑与无助。而使用该发明，用户可以通过拖拽圆来产生结点，通过选择列表中的值来确定结点，通过增加边来确定结点间的关系。

[0097] 在智能提示下，初级用户甚至可以为写出带有顺序的查询，因为只 要用户在某个结点上标注了要求有顺序限制，那么系统的智能提示功能就能反馈回当前情况下的可用值。

[0098] 3、排序的精确性

[0099] 一个好的排序公式需要考虑结构因素和内容因素来获得综合的排序结果。对于内容而言，不同的值在文档中的重要性是不同的（基于它出现的频率和分布）。对于结构而言，以下几点为重要的因素：(1) 标签权重，不同的标签需要有不同的权重，结果中包含的标签有更高的权重更可能是用户想要的，(2) 路径长度：直观地，如果在结果中的路径与在小枝中的对应的路径的分布紧凑性很接近，那么这样的结果应该有更高的分数，(3) 结点数量：一个结果中的无关结点（出现在结果中但是不出现在小枝查询中的结点）个数越多，那么这个结果很可能不是很重要。我们设计的排序公式把上面所有这些因素都涵盖进来了。

[0100] 本发明中介绍了如何开发图形化的小枝模式生成界面，并介绍了在小枝模式生成中使用到的智能提示功能的实现，包括标签名称和结点值的提示。介绍了针对查询结果的结构和内容信息进行排序的公式，并介绍了如何把结果展示为树形并为不同的结果按照排序来打分的界面。

[0101] 本发明还有许多更具有深远意义的发展前景。例如，由于本方法提出的在 XML 查询中的智能提示功能可以很容易拓展到支持 XML 的关系数据库数据库中，可以从另外一个方面优化数据库的查询方式，甚至用户可以不用学习 SQL 语言，不用对数据库有所了解就可以直接进行查询。这在各个领域都有很大的发展潜力。而且本发明的结果排序功能更是充分考虑了结构和内容信息，这无论在 XML 关键字搜索还是小枝模式查询中都有很大的应用空间。

[0102] 此外，本发明中提出的对结果的树形展示方式可以直观地让用户了解到结果的样子，这也给 XML 展示查询结果方面起了一个比较好的规范和引导作用。在支持 XML 的关系数据库中执行得到的查询结果可以在本发明的基础上进行修改，从而来更好地表示结果。

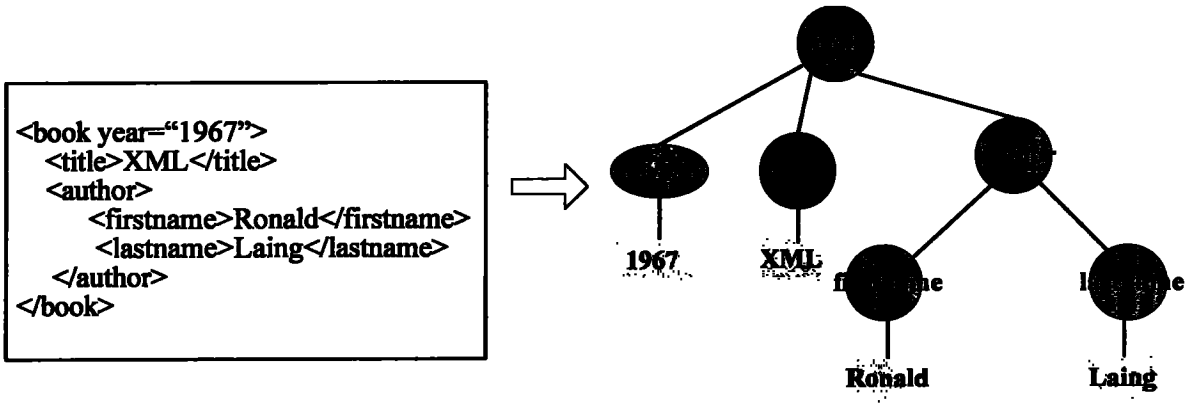

<book year="1967">
<title>XML</title>
<author>
<firstname>Ronald</firstname>
<lastname>Laing</lastname>
</author>
</book>
1967
XML
Ronald
Laing

图 1

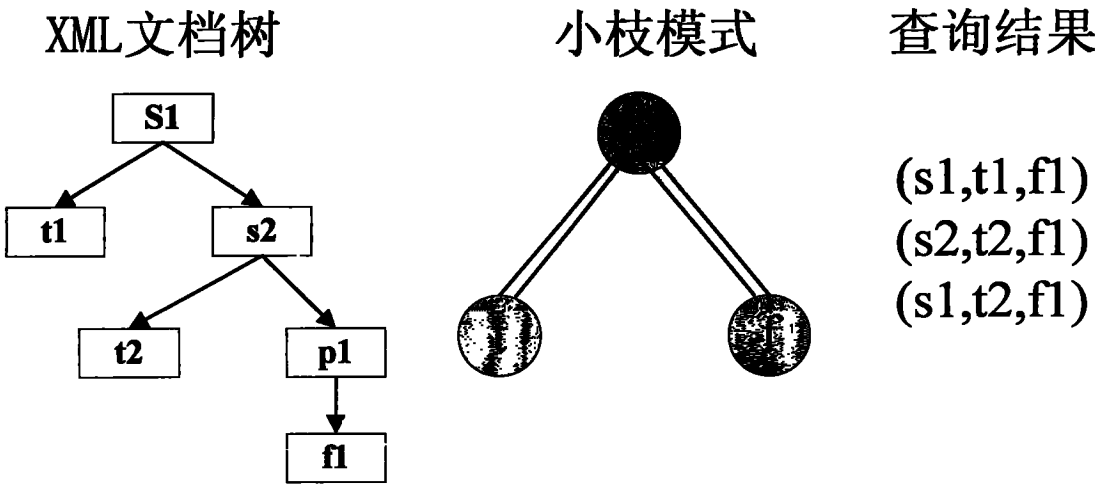

XML文档树
小枝模式
查询结果
S1
t1
s2
t2
p1
f1
(s1,t1,f1)
(s2,t2,f1)
(s1,t2,f1)

图 2

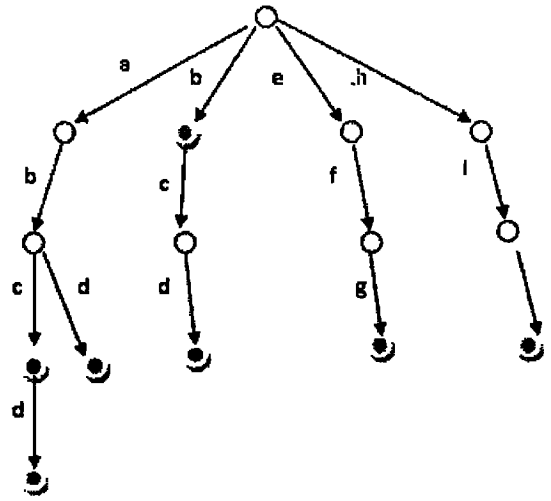

a
b
e
h
b
c
f
l
c
d
d
g
d

图 3

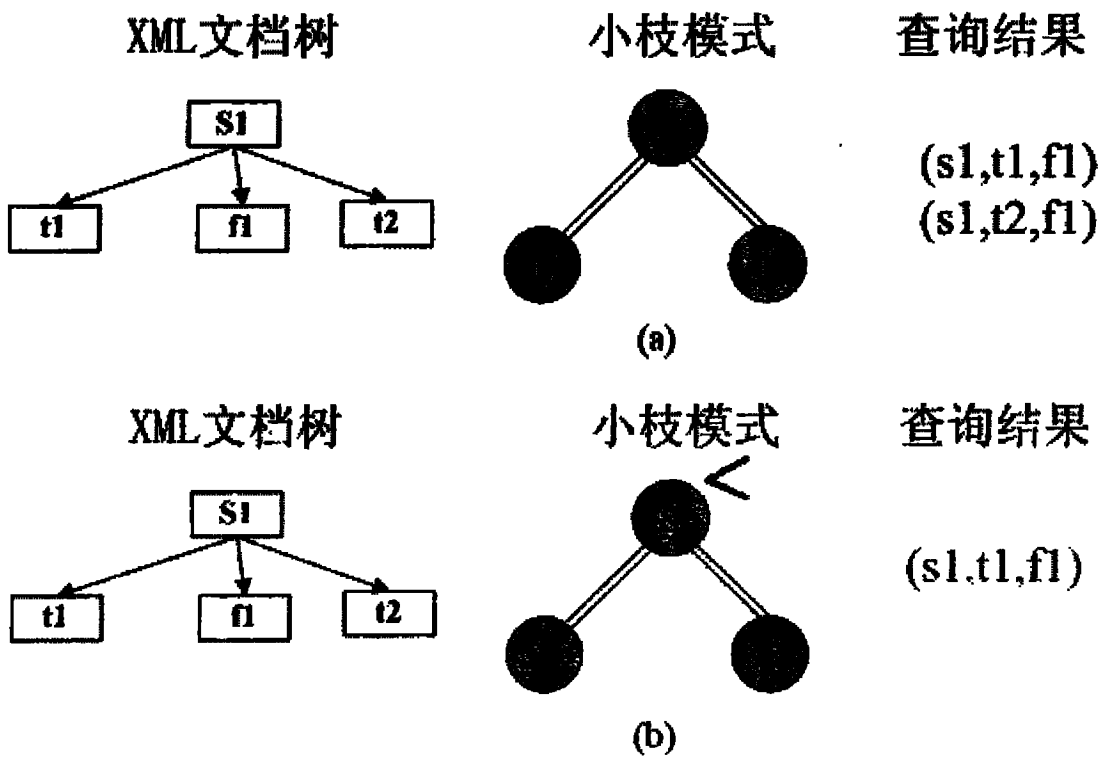

XML文档树
小枝模式
查询结果
S1
t1
f1
t2
(s1,t1,f1)
(s1,t2,f1)
(a)
XML文档树
小枝模式
查询结果
S1
t1
f1
t2
(s1,t1,f1)
(b)

图 4

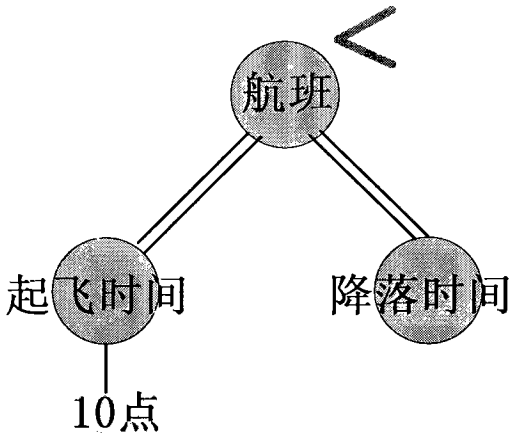

航班
起飞时间
降落时间
10点

图 5

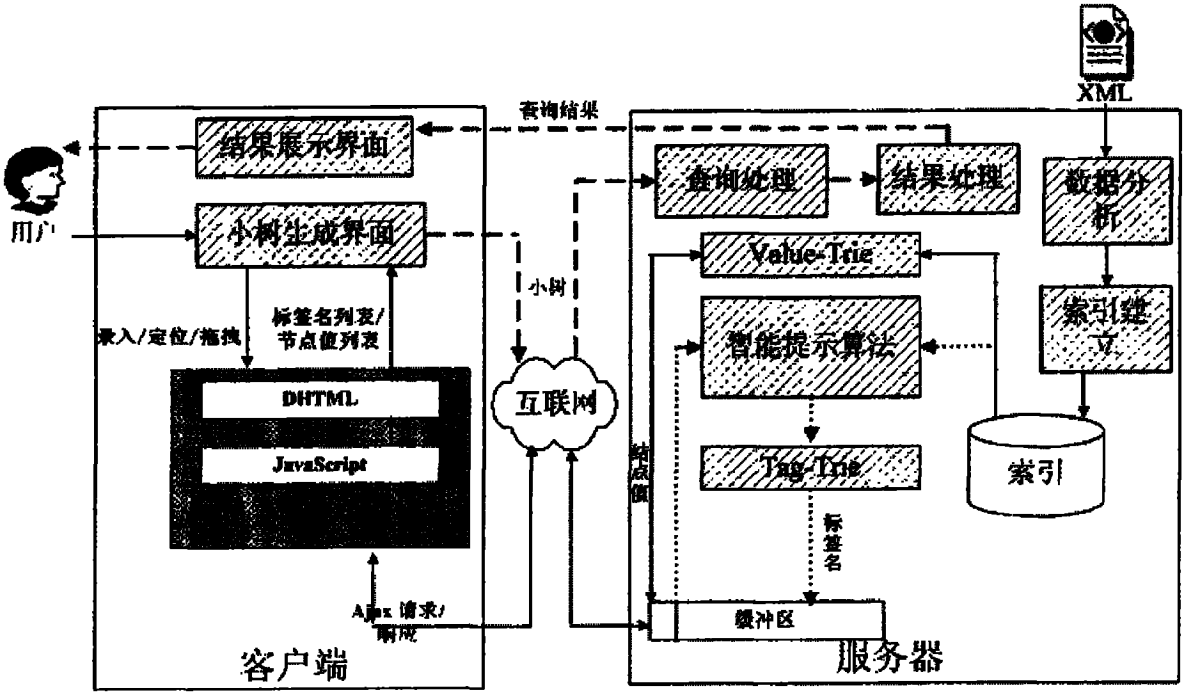

XML
查询结果
结果展示界面
用户
小树生成界面
查询处理
结果处理
数据分析
Value-Trie
小树
录入/定位/拖拽
标签名列表/节点值列表
智能提示算法
索引建立
DHTML
JavaScript
互联网
Tag-Trie
索引
结点值
标签名
Ajax 请求/响应
缓冲区
客户端
服务器

图 6

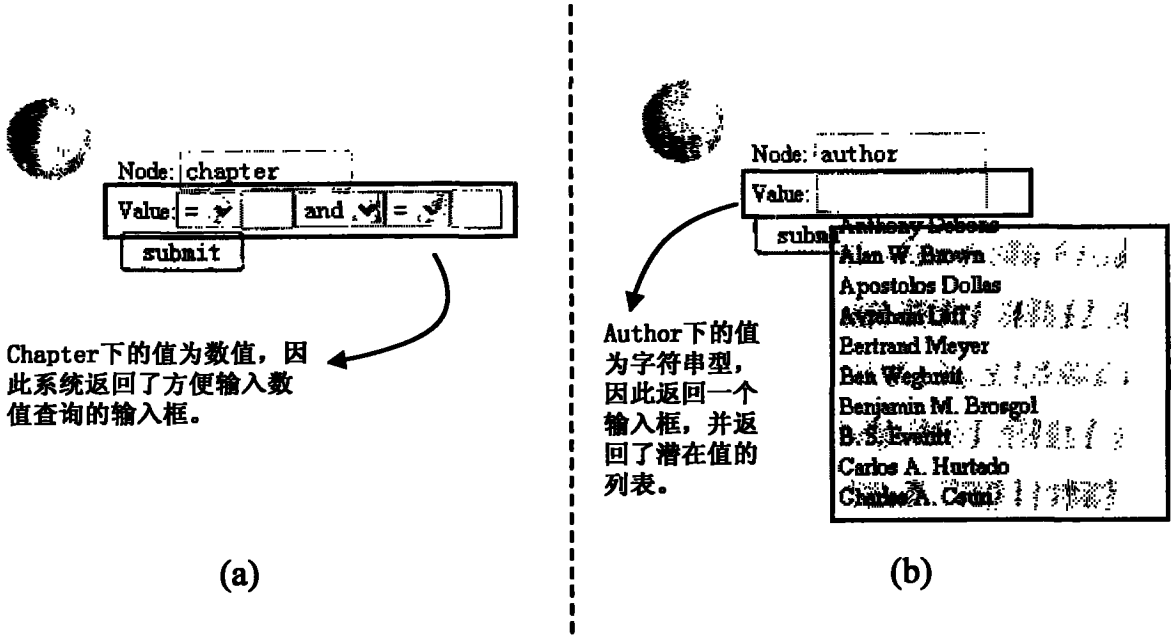

Node: chapter
Value: = and =
submit
Chapter下的值为数值，因此系统返回了方便输入数值查询的输入框。
(a)
Node: author
Value:
Anthony Debons
Alan W. Brown
Apostolos Dollas
Bertrand Meyer
Benjamin M. Brosgol
Carlos A. Hurtado
Author下的值为字符串型，因此返回一个输入框，并返回了潜在值的列表。
(b)

图 7

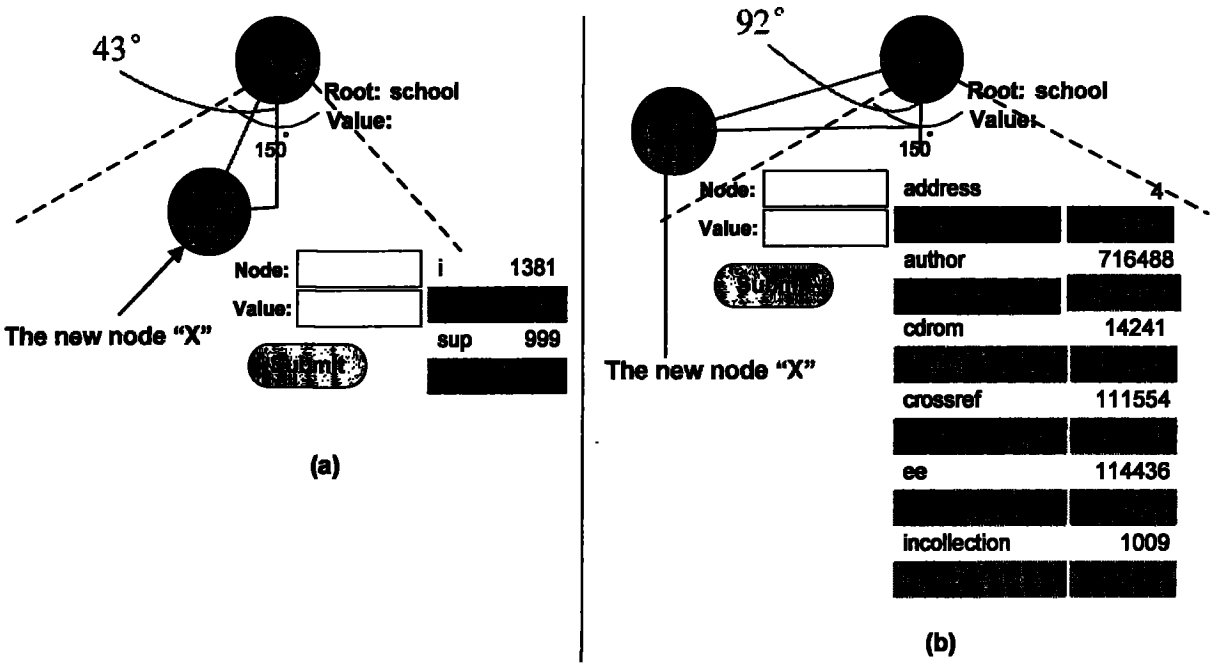

43°
Root: school
Value:
150
Node:
Value:
i 1381
sup 999
Submit
The new node "X"
(a)
92°
Root: school
Value:
150
Node:
Value:
Submit
The new node "X"
address 4
author 716488
cdrom 14241
crossref 111554
ee 114436
incollection 1009
(b)

图 8

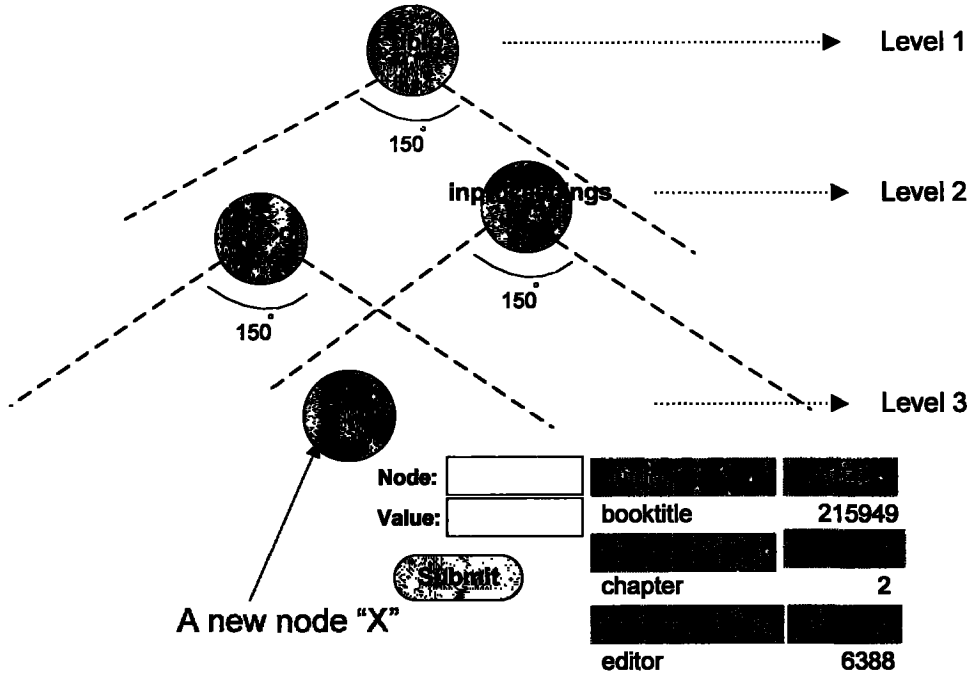

Level 1
150
Level 2
150
150
Level 3
Node:
Value:
Submit
A new node "X"
booktitle 215949
chapter 2
editor 6388

图 9

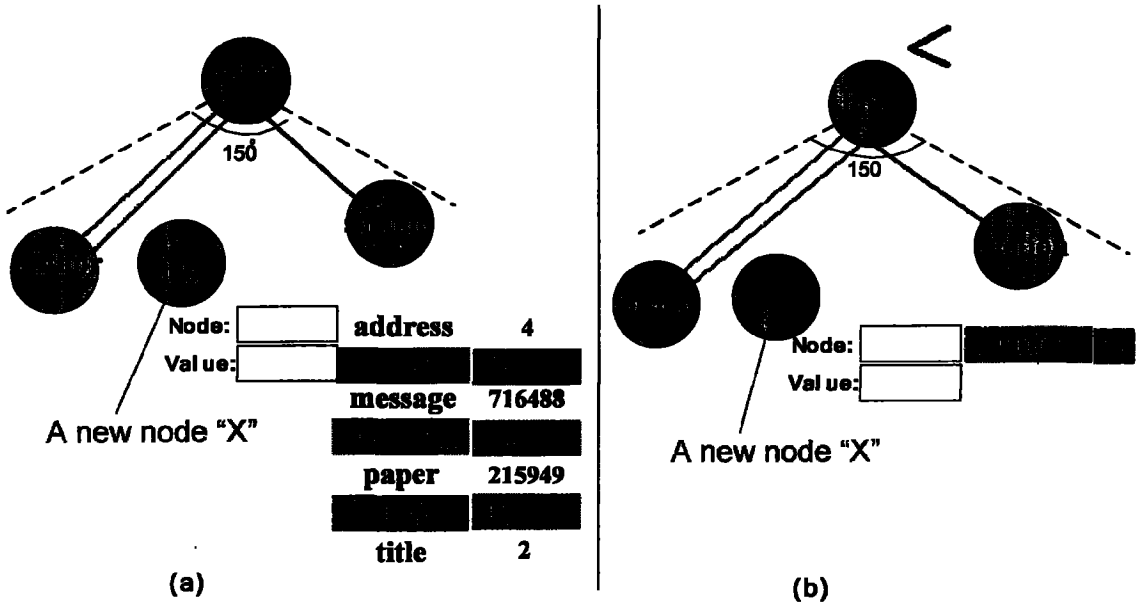

150
Node:
Value:
address
4
message
716488
paper
215949
title
2
A new node "X"
(a)
150
Node:
Value:
A new node "X"
(b)

图 10