[19] 中华人民共和国国家知识产权局

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

[12] 发明专利申请公布说明书

[21] 申请号 200780012859.2

[43] 公开日 2009 年 4 月 29 日

[11] 公开号 CN 101421729A

[22] 申请日 2007.3.1
[21] 申请号 200780012859.2
[30] 优先权
[32] 2006. 3. 3 [33] US [31] 60/778,869
[32] 2006. 5.11 [33] US [31] 11/433,139
[86] 国际申请 PCT/US2007/063105 2007.3.1
[87] 国际公布 WO2007/103749 英 2007.9.13
[85] 进入国家阶段日期 2008.10.9
[71] 申请人 奥多比公司
地址 美国加利福尼亚州
[72] 发明人 W·常 N·格哈姆拉维
A·斯沃米
[74] 专利代理机构 北京市金杜律师事务所
代理人 王茂华

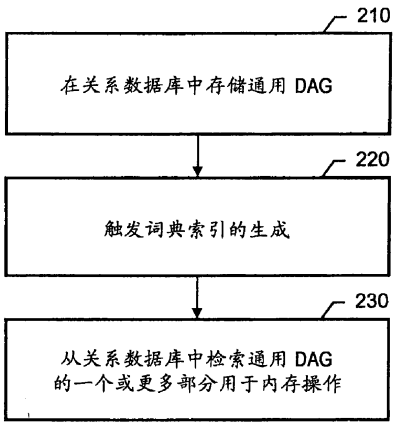
权利要求书 4 页 说明书 25 页 附图 5 页

[54] 发明名称

有效表示和搜索数据库中的有向无环图结构的系统和方法

[57] 摘要

本发明包括涉及表示和检索数据库中的数据结构的系统和技术。一般地，本发明的实施例的特征是一种计算机程序产品和方法，包括：在数据库中存储通用有向无环图(DAG)，其中存储步骤包括在数据库中的路径表的条目中编码通用 DAG 的路径信息，该编码步骤包括将路径信息转换成文本串，并且路径表中的条目对应于通用 DAG 中从通用 DAG 的节点到通用 DAG 的根节点的路径；触发使用该文本串生成路径表的词典索引，其中该词典索引分别地列出条目中包含的标记；以及从数据库中检索通用 DAG 的一个或更多部分用于内存操作。



1. 一种方法，包括:

在数据库中存储通用有向无环图（DAG），其中:

所述存储步骤包括在所述数据库中的路径表的条目中编码所述通用 DAG 的路径信息，

所述编码步骤包括将所述路径信息转换成文本串，并且

所述路径表的条目对应于所述通用 DAG 中从所述通用 DAG 的节点到所述通用 DAG 的根节点的路径;

触发使用所述文本串生成所述路径表的词典索引，其中所述词典索引分别地列出所述条目中包含的标记；以及

从所述数据库中检索所述通用 DAG 的一个或更多部分用于内存操作。

2. 根据权利要求 1 的方法，其中所述存储步骤包括在关系数据库中存储所述通用 DAG。

3. 根据权利要求 1 的方法，其中所述词典索引包括 B 树索引。

4. 根据权利要求 1 的方法，其中所述检索步骤包括使用所述词典索引查询并更新所述通用 DAG。

5. 根据权利要求 1 的方法，其中所有的条目使用指示每个相应路径的完全节点列表的标记代表相应路径。

6. 根据权利要求 1 的方法，其中所述编码步骤包括通过使用单一标记引用条目中的子路径从而压缩所述路径表。

7. 根据权利要求 6 的方法，其中所述单一标记包括代表多个子路径的占位符。

8. 根据权利要求 6 的方法，其中所述转换步骤包括根据条目格式将所述路径信息转换成文本串，所述条目格式包括说明给定路径的节点的路径标识符和路径节点列表，并且所述单一标记包括对另一条目的路径标识符的引用。

9. 根据权利要求 1 的方法，其中所述存储步骤包括在数据值节点

中分离通用 DAG 的语义单位，其中所述通用 DAG 的所述节点是引用所述数据值节点的 DAG 节点，并且每个所述数据值节点可被包括存储在所述数据库中的第二 DAG 的 DAG 节点的超过一个 DAG 节点引用。

10. 根据权利要求 9 的方法，其中所述存储步骤包括将域的多个语义刻面表示为不同的 DAG，包括所述通用 DAG 和所述第二 DAG。

11. 根据权利要求 1 的方法，其中所述存储步骤包括生成所述通用 DAG 的节点的邻接列表，所述邻接列表和所述路径表共同形成所述通用 DAG 的双重表示。

12. 根据权利要求 11 的方法，其中所述检索步骤包括使用所述路径表检索推理链和子 DAG，并且所述方法进一步包括使用所述邻接列表为所述通用 DAG 收集统计信息。

13. 一种系统，包括:

对包括图的知识结构进行编码的数据库，其中所述数据库通过将至少三个标记用于所述图中的一条路径而对所述图的路径信息进行编码，所述至少三个标记指示所述一条路径的节点;

配置用于访问所述数据库中的所述知识结构的数据库管理系统，所述数据库管理系统包括文本索引引擎; 以及

配置用于通过所述数据库管理系统存储和检索所述知识结构的信息的接口，并且所述接口配置用于触发所述文本索引引擎生成所述编码路径信息的词典索引，其中所述词典索引分别地列出所述编码路径信息中的路径的标记，包括指示所述一条路径的节点的所述至少三个标记。

14. 根据权利要求 13 的系统，其中所述词典索引包括 B 树索引。

15. 根据权利要求 13 的系统，其中所述编码路径信息包括压缩路径表，其中使用单一标记引用条目内的子路径。

16. 根据权利要求 13 的系统，其中所述知识结构包括多刻面知识结构，其包括编码的多个有向无环图(DAG)，使用 DAG 节点指示 DAG 结构以及使用数据值节点在所述多个 DAG 中共享知识。

17. 根据权利要求 13 的系统，其中所述知识结构包括邻接列表，其中所述邻接列表和编码路径信息共同形成所述图的双重表示。

18. 根据权利要求 17 的系统，其中所述接口被配置用于使用所述编码路径信息检索推理链和子 DAG，并且所述接口被配置用于使用所述邻接列表为所述图收集统计信息。

19. 一种在计算机可读介质上编码的计算机程序产品，可操作用于使数据处理装置执行操作步骤，所述操作步骤包括：

在数据库中存储通用有向无环图（DAG），其中

所述存储步骤包括在所述数据库中的路径表的条目中编码所述通用 DAG 的路径信息，

所述编码步骤包括将所述路径信息转换成文本串，以及

所述路径表的条目对应于所述通用 DAG 中从所述通用 DAG 的节点到所述通用 DAG 的根节点的路径；

触发使用所述文本串生成所述路径表的词典索引，其中所述词典索引分别地列出所述条目中包含的标记；以及

从所述数据库中检索所述通用 DAG 的一个或更多部分用于内存操作。

20. 根据权利要求 19 的计算机程序产品，其中所述存储步骤包括在关系数据库中存储所述通用 DAG。

21. 根据权利要求 19 的计算机程序产品，其中所有的条目使用指示每个相应路径的完全节点列表的标记代表所述相应路径。

22. 根据权利要求 19 的计算机程序产品，其中所述编码步骤包括通过使用单一标记引用条目中的子路径从而压缩所述路径表。

23. 根据权利要求 22 的计算机程序产品，其中所述单一标记包括代表多个子路径的占位符。

24. 根据权利要求 22 的计算机程序产品，其中所述转换步骤包括根据条目格式将所述路径信息转换成文本串，所述条目格式包括说明给定路径的节点的路径标识符和路径节点列表，并且所述单一标记包括对另一条目的路径标识符的引用。

25. 根据权利要求 19 的计算机程序产品，其中所述存储步骤包括在数据值节点中分离通用 DAG 的语义单位，其中所述通用 DAG 的所

述节点是引用所述数据值节点的 DAG 节点，并且每个所述数据值节点可被包括存储在数据库中的第二 DAG 的 DAG 节点的超过一个 DAG 节点引用。

26. 根据权利要求 25 的计算机程序产品，其中所述存储步骤包括将域的多个语义刻面表示为不同的 DAG，包括所述通用 DAG 和第二 DAG。

27. 根据权利要求 19 的计算机程序产品，其中所述存储步骤包括生成所述通用 DAG 的节点的邻接列表，所述邻接列表和所述路径表共同形成所述通用 DAG 的双重表示。

28. 根据权利要求 27 的计算机程序产品，其中所述检索步骤包括使用所述路径表检索推理链和子 DAG，并且所述操作进一步包括使用所述邻接列表为所述通用 DAG 收集统计信息。

有效表示和搜索数据库中的有向无环图结构的系统和方法

相关申请的交叉引用

本申请要求美国临时专利申请系列号 60/778,869 的优先权，该临时申请于 2006 年 3 月 3 日提交（快递邮件标签号 EV 471533919 US），名称为“SYSTEM AND METHOD OF EFFICIENTLY REPRESENTING AND SEARCHING DIRECTED ACYCLIC GRAPH STRUCTURES IN DATABASES (有效表示和搜索数据库中的有向无环图结构的系统和方法)”；并且本申请涉及美国专利申请系列号 11/368,130，该美国专利申请于 2006 年 3 月 3 日提交（快递邮件标签号 EV 471533922 US），名称为“SYSTEM AND METHOD OF BUILDING AND USING HIERARCHICAL KNOWLEDGE STRUCTURES（构建和使用层级知识结构的系统和方法）”。 上述申请通过在此引用而整体地成为本发明的组成部分。

技术领域

本发明涉及表示和检索数据库中的数据结构。

背景技术

传统地，通常以有向无环图（DAG）结构表示所存储的知识，其被称为语义网络，并且最近被称为知识本体。对于大型知识结构，DAG 结构（典型地为层级的）经常用于建立本体节点结构：弧将更多的一般（始点）节点连接到更多的特定（终点）节点。该 DAG 结构经常创建并存储于数据库中。

用于在数据库中表示树和 DAG 的传统方法包括使用邻接列表。邻接列表典型地包括成对节点，每对节点表示节点之间的父-子连接。邻接列表典型地要求导航所存储 DAG 结构，并且对于较大的图往往效率低下。插入、更新和删除效率相对较高，然而，遍历使用邻接列表存储的

本体知识结构的大型部分则效率不高。

最近的优化包括记录完整的树路径或使用所谓的嵌套集合以及区间套（nested intervals）以允许迅速有效地检索子树。这些方法一般地限于严格的树结构。使用物化路径也是一种用于表示和搜索数据库中树形数据结构的公知方法。其他技术包括分步（Fractional）方法，如 Farey 分数和连分数以及用于树结构的简单路径枚举方法。

发明内容

本说明书公开了用于表示和检索数据库中数据结构的技术的各种实施例。一般地，本发明实施例的特征是一种计算机程序产品和方法，包括：在数据库中存储通用有向无环图（DAG），其中存储步骤包括在数据库中的路径表的条目中编码通用 DAG 的路径信息，该编码步骤包括将路径信息转换成文本串，并且路径表中的条目对应于通用 DAG 中从通用 DAG 的节点到通用 DAG 的根节点的路径；触发使用该文本串生成路径表的词典索引，其中该词典索引分别地列出条目中包含的标记；以及从数据库中检索通用 DAG 的一个或更多部分用于内存操作。

这些实施例以及其他实施例能够可选地包括以下一个或更多特征。存储步骤可以包括在关系数据库中存储通用 DAG。词典索引可以包括 B 树索引，并且检索步骤可以包括使用词典索引查询并更新通用 DAG。

所有条目都可以使用指示每个相应路径的完整节点列表的标记表示相应路径。编码步骤可以包括通过使用单一标记引用条目内的子路径压缩路径表。该单一标记可以包括表示多个子路径的占位符。转换步骤可以包括根据条目格式将路径信息转换成文本串，该条目格式包括说明给定路径的节点的路径标识符和路径节点列表，并且单一标记可以包括对另一条目的路径标识符的引用。

存储步骤可以包括分离数据值节点中的通用 DAG 的语义单位，其中通用 DAG 的节点是引用数据值节点的 DAG 节点，并且每个数据值节点可被包括存储在数据库中的第二 DAG 的 DAG 节点的超过一个的 DAG 节点引用。存储步骤可以包括将域的语义的多个刻面表示为不同的

DAG，包括通用 DAG 和第二 DAG。

存储过程可以包括生成通用 DAG 节点的邻接列表，邻接列表和路径表共同形成通用 DAG 的双重表示。检索步骤可以包括使用路径表检索推理链和子 DAG，并且该方法和操作可以进一步包括使用邻接列表为通用 DAG 收集统计信息。

在另一个方面，本发明实施例的特征是一种系统，其包括对包括图的知识结构编码的系统，其中数据库通过将至少三个标记用于图中的一条路径而对图的路径信息编码，该至少三个标记指示该一条路径的节点；配置用于访问数据库中的知识结构的数据库管理系统，该数据库管理系统包括文本索引引擎；以及配置用于通过数据库管理系统存储和检索知识结构中的信息的接口，并且该接口配置用于触发文本索引引擎生成编码路径信息的词典索引，其中词典索引分别地列出编码路径信息中路径的标记，包括指示一条路径的节点的所述至少三个标记。词典索引可以包括 B 树索引，并且编码路径信息可以包括压缩的路径表，其中使用单一标记引用条目内的子路径。

知识结构可以包括多刻面知识结构，其包括多个编码的有向无环图（DAG），该多个有向无环图使用 DAG 节点指示 DAG 结构以及使用数据值节点在多个 DAG 中共享知识。知识结构可以包括邻接列表，其中邻接列表和编码路径信息共同形成图的双重表示。可以配置接口以使用编码路径信息检索推理链和子 DAG，并且可以配置接口以使用邻接列表为图收集统计信息。

可以实现本发明的特定实施例以实现以下优点中的一个或多个优点。能够在关系数据库中有效地表示 DAG，并且可以有效地检索以这种方式表示的 DAG 的子集。可以很容易地定位并访问使用本发明的系统和技术存储的知识结构的子区，并且能够实现快速推理。而且，本发明的系统和技术可以用于通用 DAG，并且由此不需要限于树图结构。本说明书中使用的词语“通用有向无环图”和“通用 DAG”指一种 DAG 结构，其中一个子节点可以有多个父节点。因此，本发明的系统和技术不需要依赖于每个子结构都有唯一可分辨父节点的知识结构。

本发明的实现可以使用通用 DAG 表示知识结构，如生成的本体。而且，DAG 可以以一种方式存储在关系数据库中，该方式一般地允许采用层级知识结构的应用有效地表示、搜索和检索 DAG。所描述的 DAG 表示系统和技术能够在知识检索和查询功能中提供显著的性能改进，并且能够通过提供一种有效定位相关证据节点、从而计算网络中任意特定事件节点的条件概率的装置，以便直接应用于大型贝叶斯信度网络。

尽管语义网络 DAG 能够提供一般框架用于表示知识、执行机器推论和推理，不过在这些结构很大时它们往往效率低下。本发明的系统和技术能够克服其他方法固有的若干效率问题。例如，本发明的系统和技术能够允许快速定位、物化和跟踪知识结构中的推理链用于机器推论应用；能够允许快速检索 DAG 子结构如主题子本体；并且能够提供一种机制用于在本体 DAG 内和本体 DAG 之间逻辑地和物理地共享知识节点值。

本发明的系统和技术可以与传统方法结合使用，如邻接列表和物化路径，并且这可以导致进一步的效率提高。与区间编码方法相比，本发明的系统和技术为非树的 DAG 结构提供了表示。并且，对于树 DAG，本发明的系统和技术可以为节点插入操作提供性能改善，并且可以要求向 DAG 路径表中增加最多 M 行，其中 M 是通过向 DAG 增加新节点而形成的新路径的总数。另外，对于树 DAG，本发明的系统和技术能够避免分数方法如 Farey 分数和连分数的可能问题，这些方法可能需要在 DAG 内节点的可能的大型子集上执行大量算术操作。本发明的系统和技术使用数字精度表示范围，能够避免可能的限制，并且在 DAG 中大约四级上不需要超出精度。对于动辄有数十级深度的大型知识结构，这具有显著优势。

本发明的系统和技术特别地可以对于本体查询操作有用，其中特别重要的查询操作可以是跟踪一系列弧，该一系列弧连接构成泛化、特化或特定推理链的节点。在推理链较长并需要以高频率执行的情况下，本发明的系统和技术能够提供显著优势。而且，本发明的系统和技术能够减少在数据库中修改知识结构时要执行的更新的数量，并且能够方便地

用于其节点可以有任意数目父节点的结构（例如其节点可以有多个父节点的本体）。

本发明的系统和技术在使用商业提供的数据库管理系统时能够提高性能。本发明的系统和技术支持使用标准查询语言如 SQL（结构化查询语言），能够方便地用于在关系数据库中表示 DAG。而且，本发明的系统和技术能够减少访问存储在数据库中的通用 DAG 所需要的子串比较的数量，并且在 DAG 的父节点的平均数量增加时能够保持有效的 DAG 表示和访问。

本发明的系统和技术支持在数据库中有效地表示通用知识 DAG 及其语义。可以改善对数据库中知识 DAG 的搜索。能够方便地跟踪并物化存储在数据库中的知识结构的推理链。能够方便地检索并且装配存储在数据库中的知识 DAG 的子图。另外，从数据/概念节点中分离结构节点并且可选地共享一些数据和结构节点，通过将域的语义的多个刻面或透视（perspective）表示为不同的 DAG，能够支持多刻面语义网络（该网络甚至可以有循环）。

附图和以下说明提供了本发明的一个或更多实施例的细节。从描述、附图和权利要求书中可以理解本发明的其他特征、方面和优势。

附图说明

图 1 是一个方框图，示出了在数据库中表示和搜索有向无环图结构的系统的例子。

图 2 是一个流程图，示出了在关系数据库中表示和搜索通用有向无环图的过程的例子。

图 3A 示出了初始有向无环图（DAG）结构的例子。

图 3B 示出了在图 3A 的 DAG 结构中增加新节点和路径的例子。

图 3C 示出了在图 3B 的 DAG 结构中增加第二新节点的效果以及结果路径的例子。

图 3D 示出了具有共享数据节点的 DAG 分类结构的例子。

图 3E 示出了具有路径压缩的 DAG 路径表的例子。

图 3F 示出了具有路径压缩的 DAG 路径表的另一个例子。

图 4 是一个流程图，示出了在关系数据库中存储通用 DAG 的过程的例子。

在各附图中相同的元件采用相同的参考标号和名称。

具体实施方式

本说明书使用的“文档”一词指电子文档。电子文档（为简洁起见简单地称为文档）不需要对应于文件。文档可以存储在保存有其他文档的文件的一部分内、在针对所考虑文档的单个文件内或在多个协调文件内。“通用有向无环图（DAG）”一词指一种 DAG 结构（有向弧并且无循环），其中一个子节点可以有多个父节点。“树”指一种 DAG 结构，其中每个节点只能有一个父节点。“图”包括树和通用 DAG。

另外，“数据库”指在长期存储器中（即在大型数据存储器中，如硬盘驱动系统，并不只在内存数据结构中）保持的综合记录的集合。“数据库管理系统”指用于定义、管理和处理数据库以及任何相关程序应用的程序集合。“关系数据库”指一种数据库，其中数据驻留在在很大程度上彼此独立的表中。

本申请描述了用于使用 DAG 表达和搜索大型知识结构的一般方法。在机器知识的结构巨大（例如相关知识条目/相互关系超过数千）时，由于种种原因，仅使用内存数据结构和算法往往变得不实用。因此可以使用基础数据库系统来管理这种知识的网络，并且本发明的系统和技术能够用于从这种使用现有数据库技术存储的知识结构中有效地表示、搜索和检索信息。

图 1 是一个方框图，示出了用于在数据库中表示和搜索有向无环图结构的系统的例子。数据库 110（例如关系数据库）对包括图的知识结构 120 编码。数据库 110 将图的路径信息编码为编码路径信息 122（例如下文所描述的 DAG 路径表）。分别使用至少三个标记对图中的一条或更多路径编码，其中该至少三个标记指示每个相应路径的节点（例如标记可以是节点标识符或弧标识符，或两者的组合）。

数据库管理系统（DBMS）150（例如关系 DBMS）被配置以访问数据库 110 中的知识结构 120。DBMS 150 能够包括文本索引引擎 160，其用于创建编码路径信息 122 的词典索引 162。DBMS 150 可以是商业提供的 DBMS（例如美国加州红杉海滨的 Oracle 公司提供的 Oracle 10g DBMS；美国纽约州白原市的国际商用机器公司提供的 IBM DB2；瑞典乌普萨拉的 MySQL AB 提供的 MySQL 服务器；或者美国华盛顿州雷蒙德的微软提供的 Microsoft SQL）。

接口 190 可配置以通过 DBMS 150 存储和检索知识结构 120 中的信息。接口 190 可以配置以触发文本索引引擎 160 生成编码路径信息 122 的词典索引 162，其中词典索引 162 分别地列出编码路径信息 122 中的路径的标记。如下文进一步描述，词典索引 162 可以包括 B 树索引，并且编码路径信息 122 可以是压缩的路径表，其中使用单一标记引用条目内的子路径。

接口 190 可以是设计与 DBMS 系统 150 操作的软件应用。接口 190 可以是单独的程序组件，或者接口 190 可以不同程度地集成到 DBMS 150，包括可能是 DBMS 150 的完全集成的程序组件。

知识结构 120 可以是包括多个编码 DAG 的多刻面知识结构，该多个 DAG 使用 DAG 节点编码指示 DAG 结构以及使用数据值节点在多个 DAG 之间共享知识。因此，可以使用 DAG 节点和数据值节点表示基础知识图，其中 DAG 节点保持图结构，但是不需要直接地对语义编码。相反，DAG 节点可以引用包含实际知识语义的适当的数据值节点。可以在 DAG 内和 DAG 之间共享数据值节点。下文结合图 3D 进一步详述其中的一个例子。DAG 和数据值节点的单独存储以多透视表示语义单位，能够促进在多个 DAG 中的概念共享。这些透视允许原子概念和概念的语义网络被整合到多个视图或知识分类刻面。

而且，知识结构 120 可以包括邻接列表 124，邻接列表 124 和编码路径信息 122 共同形成图的双重表示。接口 190 可配置以使用编码路径信息 122 检索 192 推理链和子 DAG，并且接口 190 可配置以使用邻接列表 124 为图收集 194 统计信息。该双重表示结构能够为系统提供额外

的优势。

推理是典型地通过跟踪连接相关知识节点的弧链，对知识结构中的信息泛化、特化或作出结论的过程。编码路径信息 122 能够支持快速检索推理链和子 DAG 到主存储器表示，能够显著改善这些类型操作的处理时间。而且，很好地符合邻接列表结构的其他操作能够使用邻接列表 124。邻接列表 124 可以用于插入、更新和删除维护（允许方便地执行更新），并且邻接列表 124 可以包括相关弧的权重信息，用于收集有关知识结构 120 的统计信息。

图 2 是一个流程图，示出了在关系数据库中表示和搜索通用 DAG 的过程的例子。在步骤 210，通用 DAG 可以存储在关系数据库中。该步骤可以包括在关系数据库中路径表的条目中编码通用 DAG 的路径信息，其中路径表的条目对应于通用 DAG 中从通用 DAG 的节点到通用 DAG 的根节点的路径，并且其中一个条目通过使用三个或更多标记指示路径节点来表示路径（例如标记可以是节点标识符、弧标识符或两者的结合）。

在步骤 220 可以触发生成词典索引。词典索引可以分别地列出路径表条目中包含的标记，包括指示路径节点的三个或更多标记。词典索引可以包括 B 树索引。

在步骤 230，可以从关系数据库中检索通用 DAG 的一个或更多部分用于内存操作。该检索步骤可以包括使用词典索引查询和更新通用 DAG。步骤 230 的检索步骤可以包括使用路径表检索推理链和子 DAG。而且，步骤 210 的存储步骤可以包括生成通用 DAG 的节点的邻接列表，并且在步骤 230 的检索步骤可以包括使用邻接列表为通用 DAG 收集统计信息。

图 3A-3F 示出了本发明的系统和技术的示例性实现的细节。下文对示例性实现细节的描述述及四个主要元素，用于插入节点和从 DAG 删除节点的算法，以及逻辑的和物理的数据值共享机制如何工作，该机制能够允许在单一本体和不同的本体之间共享符号知识。四个主要元素如下：（1）用于枚举 DAG 中从根节点到每个节点的所有可能路径的 DAG

路径表；（2）用于在 DAG 中迅速定位任意 DAG 节点和节点参与的所有相关路径的索引技术，该索引技术允许快速跟踪并物化推理链；（3）涉及检索和物化知识结构的子 DAG 的公共知识操作，其中路径条目可用于快速取子 DAG；以及（4）允许在 DAG 内共享 DAG 节点数据值的间接机制，并且，另外，也可以通过数据值节点在不同 DAG 之间共享数据值。

图 3A 示出了初始 DAG 结构的例子。图 3B 示出了对图 3A 的 DAG 结构增加新节点和路径的例子。图 3C 示出了对图 3B 的 DAG 结构增加第二新节点的效果以及结果路径的例子。图 3D 示出了有共享数据节点的 DAG 分类结构的例子。出于解释目的，在图 3A-3C 中，初始时未述及数据节点值共享。随后地，在图 3D 中示出并描述了数据节点共享。圆形指示的节点是 DAG 节点，即这些节点用于形成 DAG 结构。可以使用一种单独类型的数据节点用于包括数据值并且结合图 3D 讨论。

所使用的基本操作包括在 DAG 结构中插入、删除和取节点以及更新 DAG 路径表以反映这些操作。对 DAG 的操作可以通过使用以下定义的原语 A-H 的组合表示：

A. 查找从任意节点到根节点的所有路径：

查找具有特定节点作为叶的所有路径。

B. 查找根源于节点的子 DAG：

查找节点参与的所有路径，该节点作为叶的路径除外。

子 DAG 的无关部分是在特定节点之前出现的路径串的前缀。

C. 查找节点 A 和节点 B 之间的所有有向路径：

查找以任何顺序包括 A 和 B 的、以 A 和 B 其中之一为叶的所有路径。

D. 查找节点 A 和节点 B 之间所有最小长度的无向路径：

1. 查找节点 A 和节点 B 之间所有有向路径，称之为集合 Pi。
2. 查找从节点 A 到根节点所有不包括节点 B 的路径 Pa。查找节点 B 到根节点的所有不包括节点 A 的路径 Pb。取这两个集合的叉积：

$$P = Pa \times Pb$$

对于叉积 P 中的每一对节点，如果路径上最短共同前缀的长度是 k，则从两条路径的每一条上去除前 k-1 个节点。这样得出两条路径在由它们单一共同的最左端节点连接时，形成 A 和 B 之间的无向路径。

3. P 和 Pi 的合集表示节点 A 和节点 B 之间所有无向路径。

在 DAG 中插入节点包括以下原语 E 和 F:

E. 在 DAG 中插入新节点。在增加新节点时使用以下算法:

情况 1: 节点为根。

1. 如果该节点不存在，则记录节点数据。
2. 创建引用新节点的 DAG 节点 N。
3. 插入包括 DAG 节点的新路径、以及路径的叶节点 N 的标识符。

情况 2: 节点是某个其他节点 O 的子节点。

1. 如果该节点不存在，则记录节点数据。
2. 创建引用新节点的 DAG 节点 N。
3. 查找以 O 作为叶节点的所有路径。称这些路径为 Po。使 Pn 为只包括节点 N 的路径的集合。通过 Po 和 Pn 的叉积: Po × Pn，来将 Po 的所有路径连接到 Pn 的所有路径。

F. 在 DAG 中插入新弧，该弧具有始点 SRC 和终点 DST。在增加新弧时，使用以下算法:

1. 查找以 SRC 节点为叶的所有路径。称这些路径为 Pi。
2. 查找以 DST 节点为根节点的所有不同路径。称这些路径为 Pj。
3. 通过 Pi 和 Pj 的叉积: Pi × Pj，来将 Pi 的所有路径连接到所有 Pj 的路径。

如图 3A 所示，给出了知识 DAG，其节点（1）、（2）、（3）、（4）、（5）、（6）连接如图，DAG 路径表如图 3A 所示。随着增加每个节点，DAG 路径表列出到每个节点（使用叶 ID

指示为叶节点）的所有可能路径。例如对于节点（6），枚举了所有可能路径：（1）（2）（4）（6）和第二路径：（1）（5）（4）（6）。如图 3B 的例子所示，要将新节点（7）增加到 DAG。该节点的增加导致一条确切的新 DAG 路径条目：（1）（2）（7）。最后，图 3C 示出了如何创建多个 DAG 路径条目的例子。在此增加新节点（8）。指示出到节点（8）的所有可能节点：（1）（2）（4）（8）和第二路径：（1）（5）（4）（8）。

从 DAG 中删除节点涉及以下原语 G 和 H:

G. 从 DAG 中删除节点。在删除已存在节点时使用以下算法:

1. 定位该节点所在的所有路径。
2. 删除那些路径。

注意这将删除无该节点以外的祖先的所有节点。

H. 从 DAG 中删除弧。在删除已存在弧时，

1. 定位该弧所在的所有路径。
2. 删除那些路径。

注意，如果 DST 在 SRC 以外没有其他父节点，其中 SRC 和 DSC（分别地）是弧的始点和终点，则该操作等效于删除 DST。

数据节点值共享：在以上描述中，节点（4）有两个父 DAG 节点（节点（2）和节点（3））。与节点（4）相关联的概念物理地和逻辑地由其父节点共享。在语义网络中节点共享很重要；当与节点（4）相关联的概念改变时，可以在一个位置一致地完成所有更新并且确保正确地表示知识语义。并且，由于知识结构中的概念可能任意大，本发明的系统和技术的重要特征是DAG结构和包含在DAG的每个节点中的实际信息的分离。

本发明的系统和技术可以包括一种机制，通过该机制数据值可以与每个 DAG 节点关联，以便可以单独地维护和共享数据值。数据值可以是主题类、概念、一个或更多文本词或较大文本对象。在大多数情况下（例如对于知识分类和知识本体），数据值节点是共享的。另外，也可

以共享用于形成整个DAG结构的DAG节点。这可以通过下述步骤实现：将数据节点与每个 DAG 节点相关联，随后使用双向索引以确定什么数据值概念与哪一个 DAG 节点相关联，并且反过来，给定一个概念，确定使用该特定概念的所有 DAG 节点（在单一 DAG 中或在多个 DAG 之间）。

在需要共享数据值的 DAG 中，可以以如下方式扩展上文所描述的基本插入和删除算法：

有共享的 DAG 节点插入：

1. 检查是否已经存在有新数据节点值的数据节点。
2. 如果不存在匹配的数据节点，则创建一个数据节点和一个新的 ID，也创建对应的 DAG 节点和 DAG 节点的新 ID。
3. 如果找到匹配的数据节点，则获得其 ID 和相关联的 DAG 节点 ID。
4. 使用以上步骤 2 或 3 的数据节点 ID，将 DAG 节点连接到指定位置的 DAG 中。
5. 根据前文所述更新 DAG 路径表。

有共享的 DAG 节点删除：

1. 定位要删除的 DAG 节点和与其相关联的数据节点。
2. 如果所定位的 DAG 节点仅被当前 DAG 中的其他 DAG 节点引用，删除该 DAG 节点。
3. 如果所定位的 DAG 节点被其他 DAG 中的其他 DAG 引用，则仅删除来自当前 DAG 的引用 DAG 节点的链接。
4. 定位与数据节点相关联的所有 DAG 节点。
5. 如果没有对所定位的数据节点的其他引用，则删除数据节点。否则，如果有其他引用，由于其他 DAG 中的 DAG 节点仍然引用存储在当前 DAG 中的数据值，保留数据节点。
6. 根据前文所述更新 DAG 路径表。

考虑以下节点共享的例子，其中将要构建两个内容分类。第一分类按照工作角色组织人员。在一些情况下，个人可以有多个角色。第二分

类指示哪个人做哪一种产品的工作。DAG 节点以圆形表示，DAG 节点 ID 由数字表示。数据节点由矩形表示，数据节点值以文本示出。

在图 3D 的左侧，Karen、Mary 和 John 的角色是美术设计员。John、Dave、Travis 和 Karen 的角色是程序员。另外，Dave、John 和 Karen 做 Photoshop 的工作。Karen、Travis 和 Mary 做 Acrobat 的工作。两个不同的分类呈现了同一人员实体的不同视图或透视。本发明的系统和技术的一个重要方面是支持多个信息层级视图的能力，并且直接地支持刻面分类和本体的概念。

本发明的系统和技术支持用于共享 DAG 节点的两种单独的策略。每个策略都是用于构建和表示语义网络或任意 DAG 的材料。第一策略考虑仅在同一 DAG 中共享 DAG 节点。第二种策略假设可能地与其他 DAG 共享 DAG 节点。通过实施第一策略，本发明可以确保能够以 DAG 有效地表示非树 DAG。并且，可以独立于其他指向同一数据节点的其他 DAG 节点来表示 DAG 节点，保持图中其他节点之间的不同语义关系。第二策略减少创建节点的数量并且还允许 DAG 网络的互连。

图 3D 示出了两种情况下增加新元素的程序。当在有共享节点的任何 DAG 中增加新元素时，本发明的系统和技术能够定位和使用匹配的已存在值。假设存在角色的第一分类，并且要将 Dave 增加到第二分类。定位数据节点（Dave），创建相关联的 DAG 节点（12），并且将 DAG 节点（12）连接到产品分类 DAG 节点（10）。其结果是“Dave” 数据节点被角色和产品两个分类共享，但是未共享 DAG 节点（则两个分类将来可以清楚地表示关于有数据节点（Dave）的 DAG 节点的不同知识）。在图 3D 的右侧示出了该例子和两个完成的分类。在第二种情况下，与其他分类结构地共享要增加到分类的节点。假设要将 John 增加到第二分类。定位数据节点（John），定位与角色分类共享的相关联的 DAG 节点（6），并且将 DAG 节点（6）连接到产品分类 DAG 节点（10）。

假设两种类型的节点（DAG 和数据节点），并且假设每种节点类型可以是共享或非共享的，这将生成以下真值表，该表枚举了各种共享模型。

表 1

	DAG 节点	数据节点
复制子 DAG	非共享	非共享
共享概念，DAG 总是树结构	非共享	共享
未使用（意味着多个数据节点可以连接到共同的 DAG 节点）	共享	非共享
共享概念，最小存储；支持贝叶斯信度网络	共享	共享

本发明的系统和技术提供一种机制以选择适当的共享模型。默认可以总是共享 DAG 节点和数据节点。

另外，可以通过析出共同的前缀路径在 DAG 路径表中使用 DAG 路径压缩，而不是如上文描述的为每个相应路径枚举完全的节点列表。该 DAG 路径压缩可以包括通过使用路径节点列表中的路径 ID 引用子路径从而缩短路径节点列表。替代地，该 DAG 路径压缩可以包括通过占位符如“C”（尽管这可能可能地造成更高价的查询）引用任意子路径，从而缩短路径节点列表并且使 DAG 路径表中的条目更少。

参照图 3A 的图，路径条目 5 和 8 可以压缩如下。图 3E 示出了 DAG 路径表的例子，其中使用对路径节点列表中的路径 ID 的引用而实现路径压缩，而不是完全地枚举路径。因此，条目 5 的“(1)(2)(4)”由“P4 (6)”替代，其中标记“P4”指示路径节点列表的条目 4，并且标记“(6)”指示对所引用的子路径 P4 附加的节点 (6)。同样地，条目 8 的“(1)(5)(4)”由“P7 (6)”替代，其中标记“P7”指示路径节点列表的条目 7，并且标记“(6)”指示对所引用的子路径 P7 附加的节点 (6)。

该路径压缩的方法可以导致短得多的路径列表，特别是当扇出很大时（并且当有后代的节点可能有多个父节点时，并且到根有很多路径时），可得到更小的表（例如行的数量相同，串长度缩短）。然而，在这种情况下，为每个压缩的路径 ID 创建新的标记索引条目，并且额外的

复杂性可能为推理链检索和子图检索而需要多个往返操作。可以通过使用高速缓存机制存储以前曾经看到的路径 ID 及其扩展，并且随后定期地更新该高速缓存，从而实现减少这些查询数据库的额外的往返请求。

对图的更新和删除能够如上文所描述地继续操作。检索算法可以如下:

检索 DAG 节点 ID 为 N 的节点的所有后代:

1）取路径节点列表中 ID 为 N 的所有路径;

2）修整所有取得的路径节点列表，使得 N 是每个列表中的第一个节点;

3）从这些列表中检索所有节点，将其增加到后代;

4）将路径 ID 收集到 P;

5）当 P 中有路径 ID 时;

a. 检索包含 P 中的路径 ID 作为列表元素的所有路径节点列表;

b. 将 P 设置为空集;

c. 将这些所检索的列表中的节点收集到后代;

d. 将这些列表中指示的路径收集到 P;

e. 将 P 设置到所检索路径的路径 ID;

6）后代是作为 N 的后代的所有节点的集合。为了检索图的结构，使用所取得的路径或批量查询邻接列表的所有节点。为了检索节点值，查询 DAG 节点数据视图。

检索从节点 N 到根的所有路径:

1）取以 L（N）作为叶 ID 的所有路径到 S

2）将路径 ID 收集到 P

3）当 P 中有路径 ID 时

a. 检索包含 P 中的路径 ID 作为列表元素的所有路径节点列表。称之为集合 K。

b. 使用 K 重建 S 中指向 P 中路径的所有路径。

c. 重置 P，使其包含 K 的所有列表中的所有路径 ID。

4）S 是从 N 到根节点的所有路径的集合。这些路径中的 ID 可以用于从 DAG 节点数据视图查询节点名称。

图 3F 示出了 DAG 路径表的另一个例子，其中使用对压缩路径的叶节点的引用实现路径压缩（对图 3A 的图），包括路径节点列表中指示压缩路径的前缀如“C”，而不是完全地枚举路径或包括路径 ID。这与前文的路径压缩算法类似：对于路径节点列表中引用的每条路径，重写路径节点列表，以包括前缀 C 并且以路径的叶节点作为后缀，其后跟随路径节点列表中的其他节点。

由于路径 8 已经由上述路径 5 表示，去除条目 8。该表中的路径 5 表示未压缩的表中的原始路径 5（1 2 4 6）和原始路径 8（1 5 4 6）。参照检索算法，任何以 4 作为叶的路径将成为在其路径字符串中有“C”并跟随“4”作为标记的任何其他路径的前缀。因此，“C”可以视为能够表示其后的节点连接到另一条路径，并且在该例子中，跟随“C”的节点通过节点（4）连接到路径 4 和 7。

在此的检索算法可以如下：

检索 DAG 节点 ID 为 N 的节点的所有后代：

1）取路径节点列表中 ID 为 N 的所有路径；

2）修整所有取得的路径节点列表，使得 N 是每个列表中的第一个节点；

3）将所有取得的节点上的所有叶节点收集到 P；

4）从列表中检索所有节点、后代；

5）当 P 中有节点 ID 时：

a. 检索包含“C”作为第一元素并跟随有 P 中的节点 ID 的所有路径节点列表；

b. 将 P 设置为空集；

c. 将这些检索到的列表中的节点收集到后代；

d. 将这些路径上的叶节点收集到 P；

6）后代是作为 N 的后代的所有节点的集合。为了检索图的结构，使用所取得的路径或批量查询邻接列表的所有节点。为检索节

点值，查询 DAG 节点数据视图 。

检索从节点 N 到根的所有路径:

1）取以 L（N）作为叶 ID 的所有路径到 S;

2）对于 S 中以 “P” 开头的所有路径，将跟随字母 “P” 的节点 ID 收集到 P;

3）当 P 中有节点 ID 时:

a. 检索包含 P 中的节点 ID 作为叶节点的所有路径节点列表。称之为集合 K;

b. 使用 K 重建 S 中引用 P 中路径的所有路径;

c. 重置 P，使其包含 K 中所有列表中的所有路径 ID;

4）S 是从 N 到根节点的所有路径的集合。这些路径中的 ID 可以用于从 DAG 节点数据视图中查询节点名称。

该第二方法能够导致对路径表更大程度的压缩。然而，该第二方法也可能使得查询更高价或需要更多过滤：例如，在检索子图的步骤 5a 中，施加了对路径节点列表第二 “元素” 的要求，这是比查询单个（索引的）标记更高价的查询操作。在上述两个路径压缩的例子中，只有图中选择的路径被压缩：叶节点有多个父节点的那些路径。因此，只有图 3A 的原始表中的路径 5 和 8 被压缩。

如前文所述，使用的基本操作是在 DAG 结构中插入、删除和取节点并且更新 DAG 路径表以反映这些操作。对 DAG 的所有操作可以使用以下定义的原语 A-H 的组合表示:

A. 查找节点 A 和节点 B 之间的所有有向路径:

查找以任何顺序包括 A 和 B 的、以 A 和 B 之一为叶的所有路径;

查找以 A 为叶的所有路径和一些压缩路径；检索被引用路径中包含 B 的路径;

查找以 B 为叶的所有路径和一些压缩路径；检索被引用路径中包含 A 的路径。

B. 查找节点 A 和节点 B 之间所有最小长度的无向路径(算法不变):

1. 查找节点 A 和节点 B 之间的所有有向路径，称之为集合 Pi。
2. 查找从节点 A 到根节点所有不包括节点 B 的路径 Pa。查找节点 B 到根节点所有不包括节点 A 的路径 Pb。取这两个集合的叉积:

$$P = Pa \times Pb$$

对于叉积 P 中的每一对节点，如果路径上最短共同前缀的长度是 k，从两条路径的每一条上去除前 k-1 个节点。这样得出两条路径，在由它们单一共同的最左端节点连接时，形成 A 和 B 之间的无向路径。

3. 注意，叉积可能去除任何所得路径上的冗余节点（例如图中以 C 为根的 A 和 D 之间的路径是 A、B、C、B、D，但是最短路径仅是 A、B、D）。
4. P 和 Pi 的合集代表节点 A 和节点 B 之间所有的无向路径。

插入算法

C. 在 DAG 中插入新节点。在增加新节点时可以使用以下算法（在未压缩的情况下这两种算法是相同的，因为压缩路径引用总是跟随着至少一个节点 ID，因此例如查找以 O 为叶节点的所有路径的任务是不变的):

情况 1: 节点为根

1. 如果该节点不存在，则记录节点数据。
2. 创建引用新节点的 DAG 节点 N。
3. 插入包括 DAG 节点的新路径、以及路径的叶节点 N 的标识符。

情况 2: 节点是某个其他节点 O 的子节点。

1. 如果该节点不存在，则记录节点数据。
2. 创建引用新节点的 DAG 节点 N。
3. 查找以 O 作为叶节点的所有路径（仅行）。称这些路径为 Po。

使 Pn 为只包括节点 N 的路径的集合。通过 Po 和 Pn 的叉积：Po × Pn，来将 Po 的所有路径连接到 Pn 的所有路径。

D. 在 DAG 中插入新弧，该弧具有始点 SRC 和终点 DST。在增加新弧时，使用以下算法：

1. 查找以 SRC 节点为叶的所有路径。称这些路径为 Pi。

a. 这类似于查找从 SRC 到根的所有路径、或所有结束于 SRC 的“推理链”。

2. 查找以 DST 节点为根节点的所有不同路径。称这些路径为 Pj。

a. 这完全地类似于查找 DST 的所有后代：首先查找所有包含 DST 的所有路径，并且随后也查找所有引用任何这些路径的压缩路径。在第一种压缩方法中，这通过搜索引用所考虑路径 ID 的压缩路径实现；在第二种压缩方法中，这等于查找所有以 P 开头跟随所考虑节点 ID 的所有压缩路径（所检索到的路径之一的叶节点）。

3. 通过 Pi 和 Pj 的叉积：Pi × Pj，来将 Pi 的所有路径连接到所有 Pj 的路径。

删除算法

E. 从 DAG 中删除节点。在删除已存在节点时使用以下算法：

1. 定位该节点所在的所有路径：

a. 在压缩情况下，这等于也定位引用/压缩路径：首先是包含节点的任何路径，随后是引用已存在路径的任何路径（并且重复该过程直到不再有引用的新路径）。

2. 删除那些路径。

注意，这将删除无该节点以外的祖先的所有节点。

H. 从 DAG 中删除弧。在删除已存在弧时使用以下算法：

1. 定位该弧所在的所有路径。

2. 删除那些路径。

注意，如果 DST 在 SRC 以外没有其他父节点，则其中 SRC 和 DSC（分别地）是弧的始点和终点，则该操作等效于删除 DST。

在路径被压缩时该算法的情况稍微有些复杂:

在使用路径 ID 的第一种压缩算法中，这等于查找包含由 DST 跟随的 SRC 的所有路径，以及以 DST 作为第一非压缩节点并且引用以 SRC 为叶的路径的所有压缩路径，并且删除那些路径。

在第二种算法中，由于没有明确地提及的路径并且在路径表中明确地捕获了所有弧，因此删除弧的算法不变。

语义网络中 DAG 的典型使用适用于大部分为树的 DAG。也就是说，大部分节点只有一个父节点，而有些节点可能有更多父节点，但是可能地少于三个。因此，需要表示通用 DAG 的数据结构，同样地优选对于近乎为树的 DAG 最方便的该数据结构的实现。上文描述的路径压缩方法一般很好地适合这点。因为在很多实现中典型的 DAG 大部分是树，第二种压缩方法相对于第一种压缩方法的优势可能很小。而且，由于数据库标记索引方法可能不解释标记的序数值，因而第二种压缩方法可能涉及更高价的查询。因此在 DAG 大部分为树的实现中，相对于第二种压缩方法可能优选第一种压缩方法。

另外，应当注意，路径压缩提供几种可能的空间节约优势。重要的考虑因素包括何时压缩路径（在任何更新时或在离线批量模式）以及有关压缩哪些路径的决策。对于第二个考虑因素，可以压缩 DAG 中的所有路径，使得，例如，路径节点表永远不会超过两个节点 ID，并且永远有压缩路径（除了在根上）。然而，由于这倾向于否定本说明书所描述的路径枚举和词典索引的优势，可能并不是所期望的。考虑到这点，可以根据图的种类和期望对图所做的更新的种类对压缩做适当的试探，以便优化查询和所需的空间。

一个试探是，总是压缩有至少四个节点的路径。另一个试探是，根据入度（in-degree）和出度（out-degree）压缩。例如，节点的两个孩子共享节点到根节点所有相同路径，因此路径压缩的良好候选试探是在有

超过四个孩子的节点上。如果那些孩子有后代，则可能是路径压缩更好的候选。另一个重要考虑因素是，在有超过一个父节点的节点上压缩路径。

路径压缩可以与插入和删除算法整合以确保实行对路径长度和深度以及入度的约束。替代地，路径压缩可以在离线时以批量模式应用以确保实行约束。路径压缩可以是两种方法的结合：例如，考虑将子图作为某个其他节点 N 的孩子插入；可以在线压缩从 N 到根的路径。

在很多实现中，图大部分地于扩增过程中在内存中创建，或通过增加多个节点子图扩增。因此，对整个 DAG 的路径压缩可以延迟为批处理模式，或在 DAG 存续期间进行，以便允许最佳的路径压缩。对于本说明书描述的 DAG，期望的是，更新（弧和节点的插入和删除）将可能随着时间缓慢地改变图，因此除非插入多节点子图，可以用批量处理模式压缩。而且，由于在很多实现中使用的图大部分是树，优选地使用第一种压缩方法，并且为有四个或更多节点的路径以及叶节点有至少四个出度或至少两个入度的路径压缩路径。

图 4 是流程图，示出了在关系数据库中存储通用 DAG 的过程的例子。在步骤 410，可以如上文所描述，根据条目格式将路径信息转换成文本串。在步骤 420，如上文所描述，可以通过使用单一标记引用条目内的子路径从而压缩路径表。步骤 410 的转换和步骤 420 的压缩步骤代表上文描述的编码过程的操作并且不需要是分别的操作。

在步骤 430，如上文所描述，可以在数据值节点中分离 DAG 的语义单位。而且，如上文所描述，可以在步骤 440 生成 DAG 节点的邻接列表以创建 DAG 的双重表示。

本发明的实施例和本说明书所描述的全部功能性操作可以以包括在本说明书中公开的结构及其对等结构的数字电子电路、计算机软件、固件或硬件的形式实现，或以以上的一个或更多的组合的形式实现。本发明的实施例可以作为一个或更多计算机程序产品实现，即一个或更多计算机程序指令模块，其编码于计算机可读介质上用于由数据处理装置执行或用于控制数据处理装置的操作。计算机可读介质可以是机器可读

的存储装置、机器可读的存储基体、存储器装置、实现机器可读传播信号的物质组合、或以上一个或更多的组合。术语“数据处理装置”包括所有用于处理数据的装置、设备和机器，包括例如可编程处理器、计算机或多个处理器或计算机。除了硬件，装置可以包括为所考虑计算机程序创建执行环境的代码，例如构成处理器固件、协议堆栈、数据库管理系统、操作系统或以上一个或更多的组合的代码。传播信号是人工生成的信号，例如机器生成的电、光或电磁信号，其生成以编码信息用于传输给适当的接收器装置。

可以以任何形式的编程语言（包括编译或解释语言）编写计算机程序（也公知为程序、软件、软件应用、脚本或代码），并且可以以任何形式布置，包括作为独立程序或作为模块、组件、子程序或其他适用于在计算环境中使用的单元。计算机程序不需要对应于文件系统中的文件。程序可以存储在文件的一个部分，该文件保存其他程序或文件（例如存储于标记语言文档中的一个或更多脚本），存储在针对所考虑程序的单一文件中，或存储在多个协调文件中（例如存储一个或更多模块、子程序或代码部分的文件）。计算机程序可以布置为在一台计算机或多台计算机上执行，该多台计算机位于一个地点或分布在多个地点并且通过通信网络互联。

本说明书中描述的过程和逻辑流程可以由一个或更多可编程处理器执行，该可编程处理器执行一个或更多计算机程序，以通过对输入数据操作和生成输入执行功能。过程和逻辑流程也可以由专用逻辑电路执行，并且装置也可以作为专用逻辑电路实现，例如 FPGA（现场可编程门阵列）或 ASIC（专用集成电路）。

适用于执行计算机程序的处理器包括，例如，通用和专用微处理器，以及任何种类数字计算机的任意一个或更多处理器。一般地，处理器从只读存储器或随机存取存储器或从两者接收指令和数据。计算机的基本元件是用于执行指令的处理器以及用于存储指令和数据的一个或更多存储装置。一般地，计算机也包括或可操作地连接到用于存储数据的一个或更多大型存储装置，如磁盘、磁-光盘或光盘，以便从其接收数据或

向其发送数据或既接收也发送数据。然而，计算机不需要有这样的装置。而且，计算机可以嵌入到另一个装置中，例如移动电话、个人数字助理（PDA）、移动音频播放器、全球定位系统（GPS）接收器，等等。适用于存储计算机程序指令和数据的信息载体包括所有形式的非易失性存储器、介质和存储装置，包括：例如半导体存储装置，例如 EPROM、EEPROM 以及闪存装置；磁盘，例如内部硬盘或可移除磁盘；磁-光盘；以及 CD-ROM 和 DVD-ROM 盘。处理器和存储器可以由专用逻辑电路补充或集成于其中。

为了提供与用户的交互，本发明的实施例可以在有显示装置（例如 CRT（阴极射线管）或 LCD（液晶显示器）显示器）的计算机上实现，用于向用户显示信息，并且计算机有键盘和指针设备，例如鼠标或轨迹球，用户可以通过这些向计算机提供输入。也可以使用其他种类的装置提供与用户的交互；例如，提供给用户的反馈可以是任何形式的感觉反馈，例如视觉反馈、听觉反馈或触觉反馈；并且可以以任何形式接收来自用户的输入，包括声音、语音或触觉输入。

本发明的实施例可以在计算系统中实现，该计算系统包括后端组件，例如作为数据服务器；或包括中间件，例如应用服务器；或包括前端组件，例如客户端计算机，客户端计算机有图形用户接口或网络浏览器，用户通过这些可以与本发明的实现交互；或包括这种后端组件、中间件或前端组件的任何组合。系统的组件可以由例如通信网络的数字数据通信的任何形式或介质互联。通信网络的例子包括局域网（“LAN”）和广域网（“WAN”），例如因特网。

计算系统可以包括客户端和服务器。客户端和服务器一般彼此距离遥远并且典型地通过通信网络交互。客户端和服务器的关系是通过计算机程序实现的，这些计算机程序在相应计算机上运行，并且彼此存在客户端-服务器关系。

尽管本说明书包括很多细节，但这些不应理解为对本发明范围或权利要求书的限制，而是对特定于本发明特定实施例的特征的描述。本说明书中描述的在不同实施例环境下的某些特征也可以在单一实施例中

组合实现。反过来，所描述的在单一实施例环境下的各种特征也可以在多个实施例中分别地实现或以任何适当的子组合实现。而且，尽管上文中的特征描述为以某些组合作用并且甚至初始地在权利要求书中如此描述，在某些情况下，权利要求书中的组合的一个或更多特征可以从组合中分离，并且权利要求书中的组合可以用于子组合或子组合的变型。

类似地，尽管在图中以特定的顺序示出操作，但不应理解为要求以示出的特定顺序或顺序地执行这些操作，或者要执行所有示出的操作，以实现所期望的结果。在某些情况下，多任务处理和平行处理可能是有利的。而且，上文描述的实施例的各种系统组件的分离不应理解为在所有实施例中都需要这种分离，并且应当理解，所描述的程序组件和系统能够一般地共同集成到单一软件产品或打包到多个软件产品中。

由此已经描述了本发明的特定实施例。其他实施例也在所附权利要求书的范围内。例如，可以以不同的顺序执行权利要求书中陈述的动作，并且仍旧实现所期望的结果。

数据库表可以存储在文件中（行和制表符分隔）或以哈希表存储在内存中。索引可以是内存 B 树或哈希表。增加表的行带来文件中行的增加或查询适当的哈希表并且执行表和索引更新。查询可以包括扫描文件并且执行常规表达模式匹配，并且类似地查询数据结构。另外，数据库表中可以包括更多信息。例如，路径表也可以包括 DAG 弧的信息（例如弧的权重，弧的类型等），如通过增加对应于路径表中的路径节点列表的路径弧列表。也可以以对应于路径节点列表的方式处理路径弧列表，包括对路径弧列表使用词典索引和压缩。

而且，本发明的系统和技术可以在比仅关系数据库更广泛的环境下实现，如使用文件系统、B 树、关联存储器或面向对象的数据库（OODB）。例如，本发明可以通过定义封装 DAG、节点、弧和路径对象及其行为（通过适当的基于索引的路径访问）的 OODB 复杂对象实现，在这种情况下，本发明可以直接使用商业 OODB 产品，如 Objectivity/DB（美国加州桑尼维尔的 Objectivity 公司提供）、Ontos DB（美国马萨诸塞州罗尼尔的 Ontos 公司提供）、Versant（美国加州门洛帕克的 Versant Object

Technology 公司提供)、ObjectStore(美国马萨诸塞州伯灵顿的 Object Design 公司提供)、或 GemStone(美国俄勒冈州比弗顿的 GemSton Systems 公司提供)。

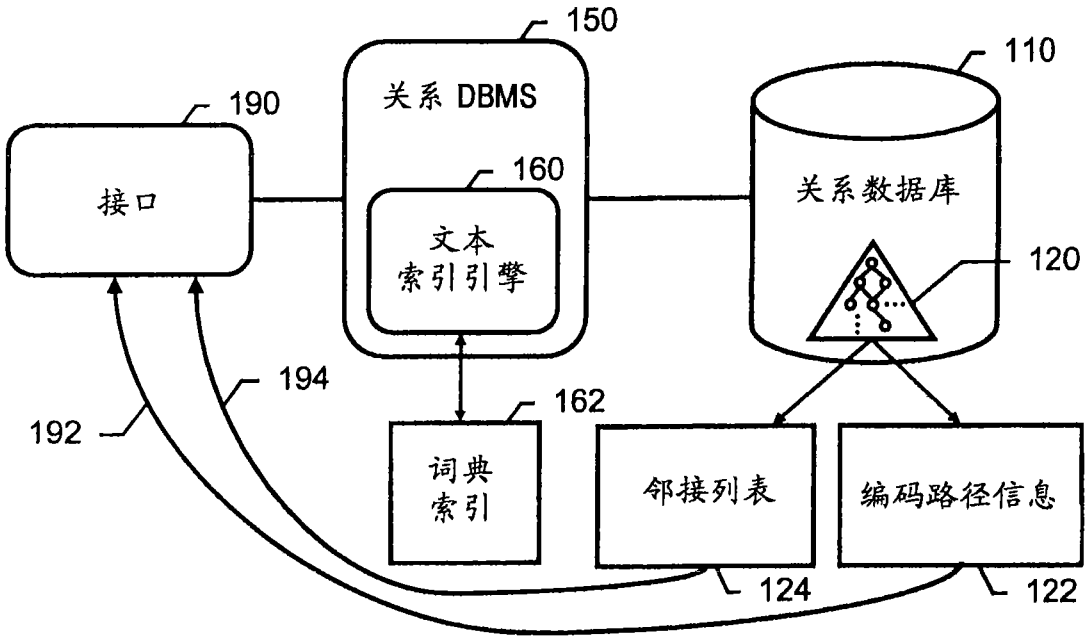

150
关系 DBMS
160
文本
索引引擎
190
接口
110
关系数据库
120
194
192
162
词典
索引
邻接列表
编码路径信息
124
122

图 1

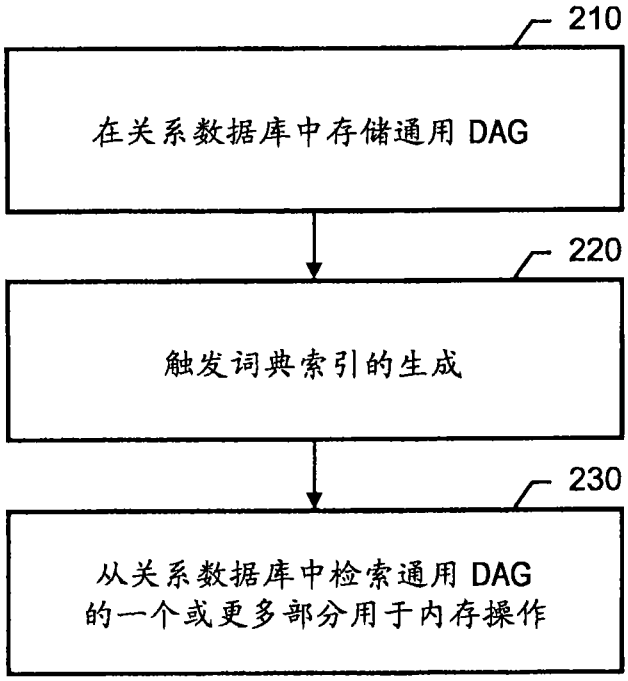
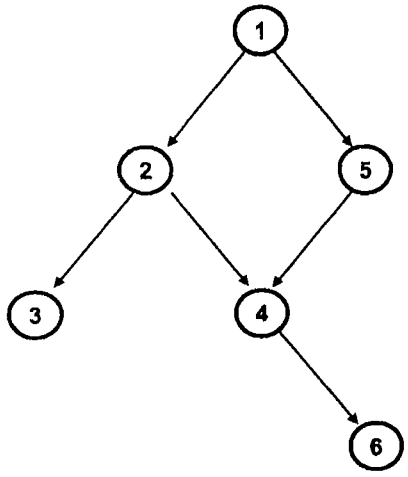

210
在关系数据库中存储通用 DAG
220
触发词典索引的生成
230
从关系数据库中检索通用 DAG
的一个或更多部分用于内存操作

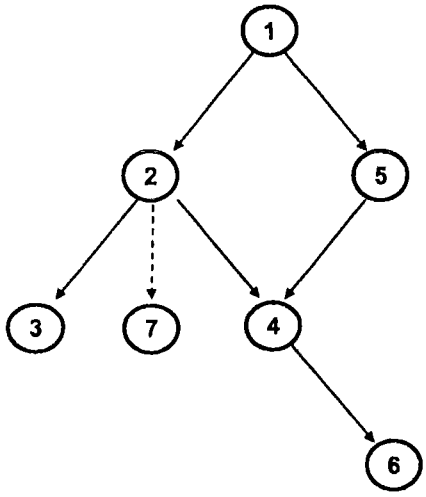
图 2



DAG 路径表

路径 ID	叶 ID	路径节点列表
1	L(1)	(1)
2	L(2)	(1) (2)
3	L(3)	(1) (2) (3)
4	L(4)	(1) (2) (4)
5	L(6)	(1) (2) (4) (6)
6	L(5)	(1) (5)
7	L(4)	(1) (5) (4)
8	L(6)	(1) (5) (4) (6)

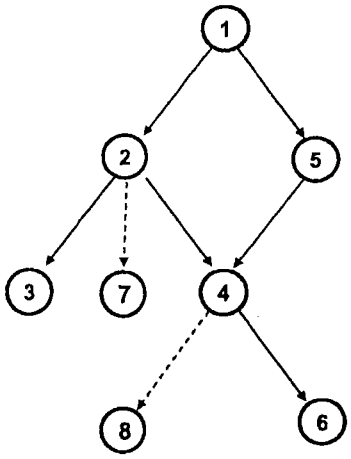
图 3A



DAG 路径表

路径 ID	叶 ID	路径节点列表
1	L(1)	(1)
2	L(2)	(1) (2)
3	L(3)	(1) (2) (3)
4	L(4)	(1) (2) (4)
5	L(6)	(1) (2) (4) (6)
6	L(5)	(1) (5)
7	L(4)	(1) (5) (4)
8	L(6)	(1) (5) (4) (6)
9	**L(7)**	**(1) (2) (7)**

图 3B



DAG 路径表

路径 ID	叶 ID	路径节点列表
1	L(1)	(1)
2	L(2)	(1) (2)
3	L(3)	(1) (2) (3)
4	L(4)	(1) (2) (4)
5	L(6)	(1) (2) (4) (6)
6	L(5)	(1) (5)
7	L(4)	(1) (5) (4)
8	L(6)	(1) (5) (4) (6)
9	L(7)	(1) (2) (7)

路径 ID	叶 ID	路径节点列表
10	L(8)	(1) (2) (4) (8)
11	L(8)	(1) (5) (4) (8)

图 3C

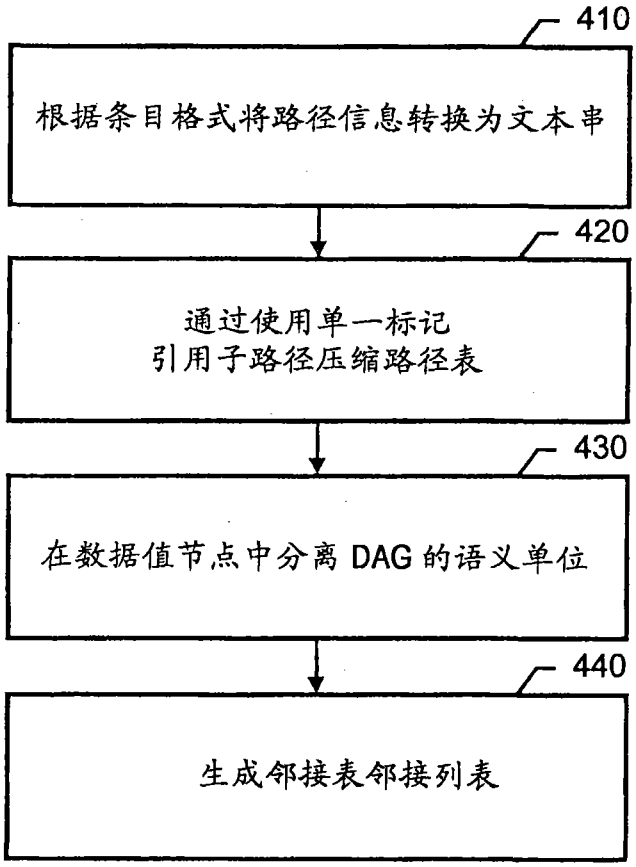


图 4

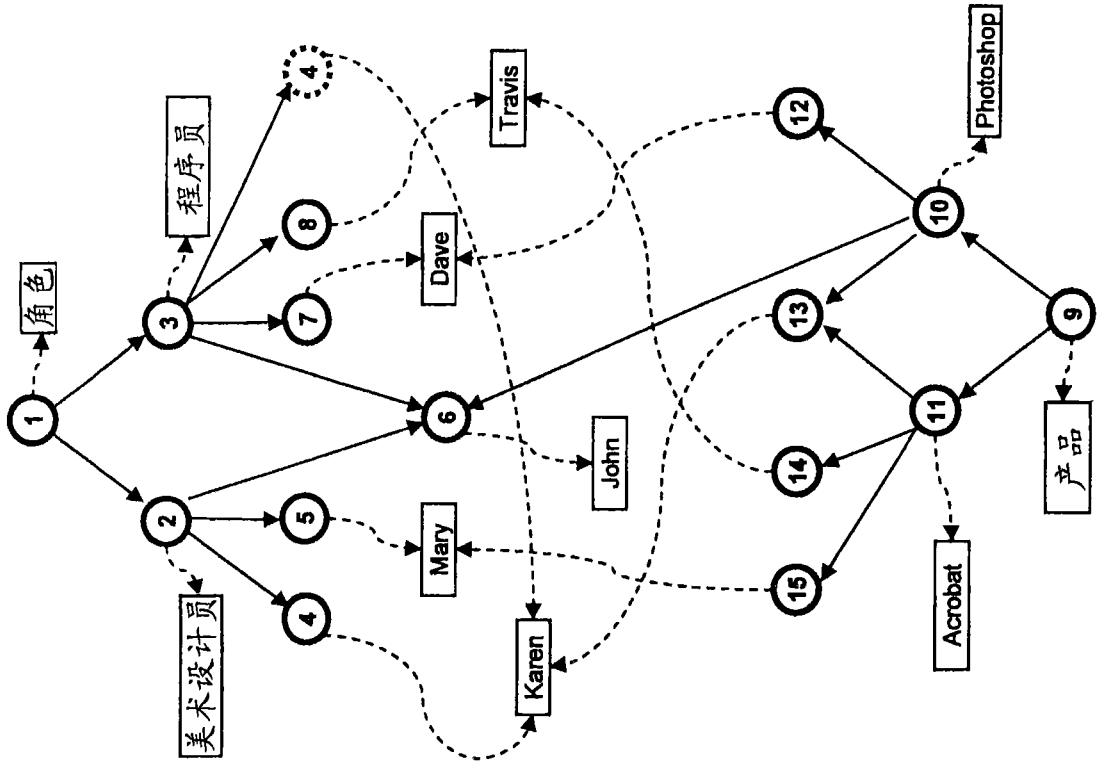
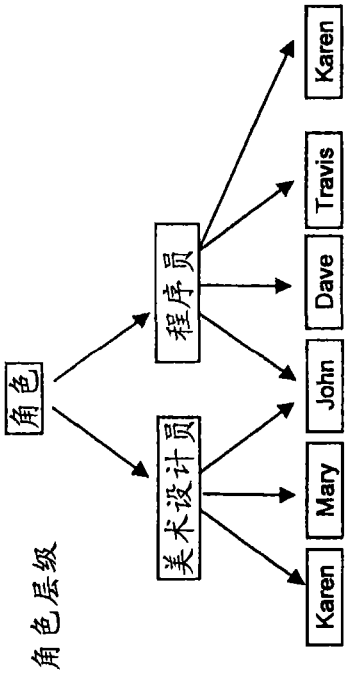
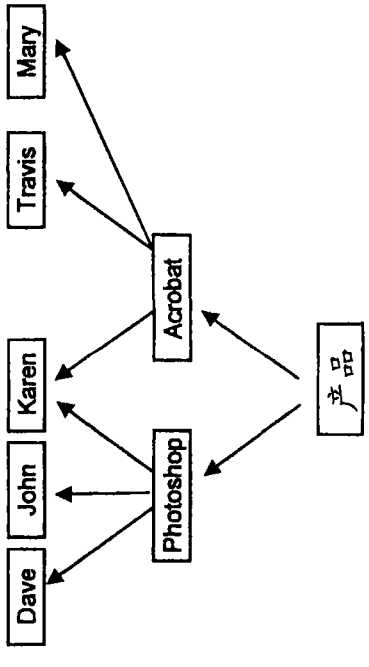

角色
程序员
美术设计员
Dave
Travis
John
Mary
Karen
Photoshop
Acrobat
产品
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

图 3D


角色层级
角色
美术设计员
程序员
Karen
Mary
John
Dave
Travis
Karen


产品层级
产品
Photoshop
Acrobat
Dave
John
Karen
Travis
Mary

路径 ID	叶 ID	路径节点列表
1	L(1)	(1)
2	L(2)	(1) (2)
3	L(3)	(1) (2) (3)
4	L(4)	(1) (2) (4)
5	**L(6)**	**P4 (6)**
6	L(5)	(1) (5)
7	L(4)	(1) (5) (4)
8	**L(6)**	**P7 (6)**

图 3E

路径 ID	叶 ID	路径节点列表
1	L(1)	(1)
2	L(2)	(1) (2)
3	L(3)	(1) (2) (3)
4	L(4)	(1) (2) (4)
5	**L(6)**	**C (4) (6)**
6	L(5)	(1) (5)
7	L(4)	(1) (5) (4)

图 3F