

《知识图谱: 概念与技术》

第 10 讲 查询与检索

郑卫国

香港中文大学

wgzheng@se.cuhk.edu.hk

本章大纲

- 知识图谱查询概述
- 查询语言：SPARQL
- SPARQL查询
- 子结构查询
- 关键字查询
- 其他查询

知识图谱查询概述

知识图谱查询概述

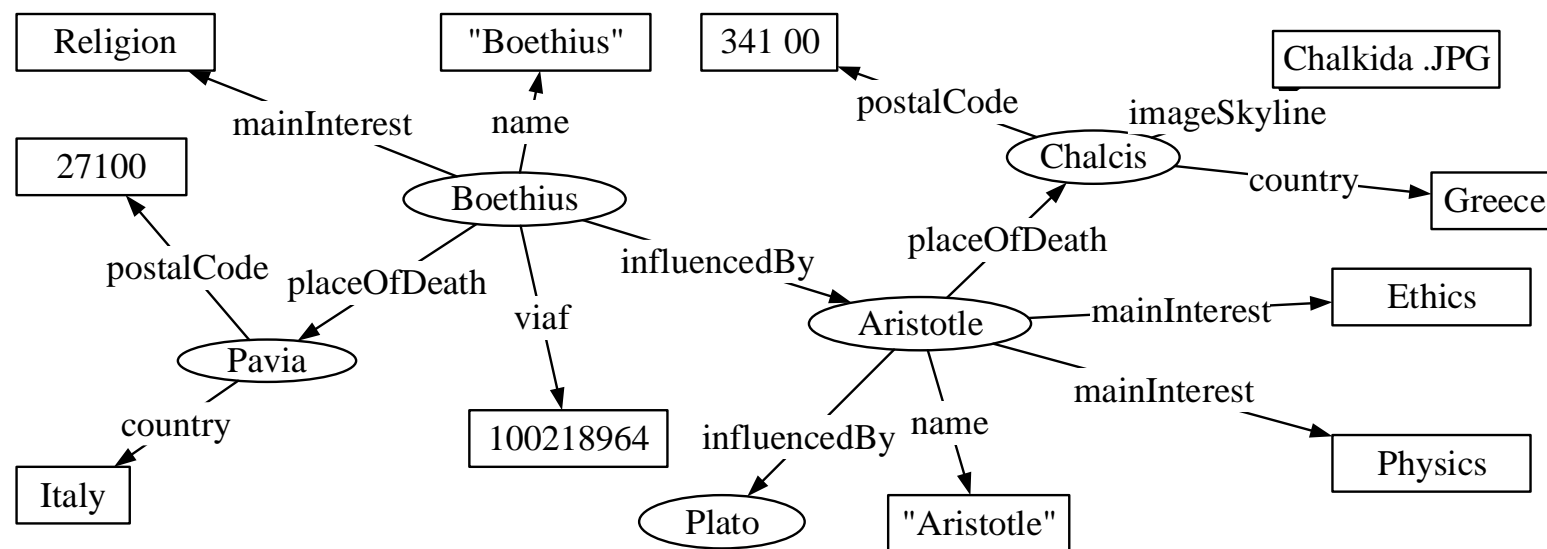
- 知识图谱

- 语义网络
- 点：实体、概念
- 边：关系、属性

- 本质

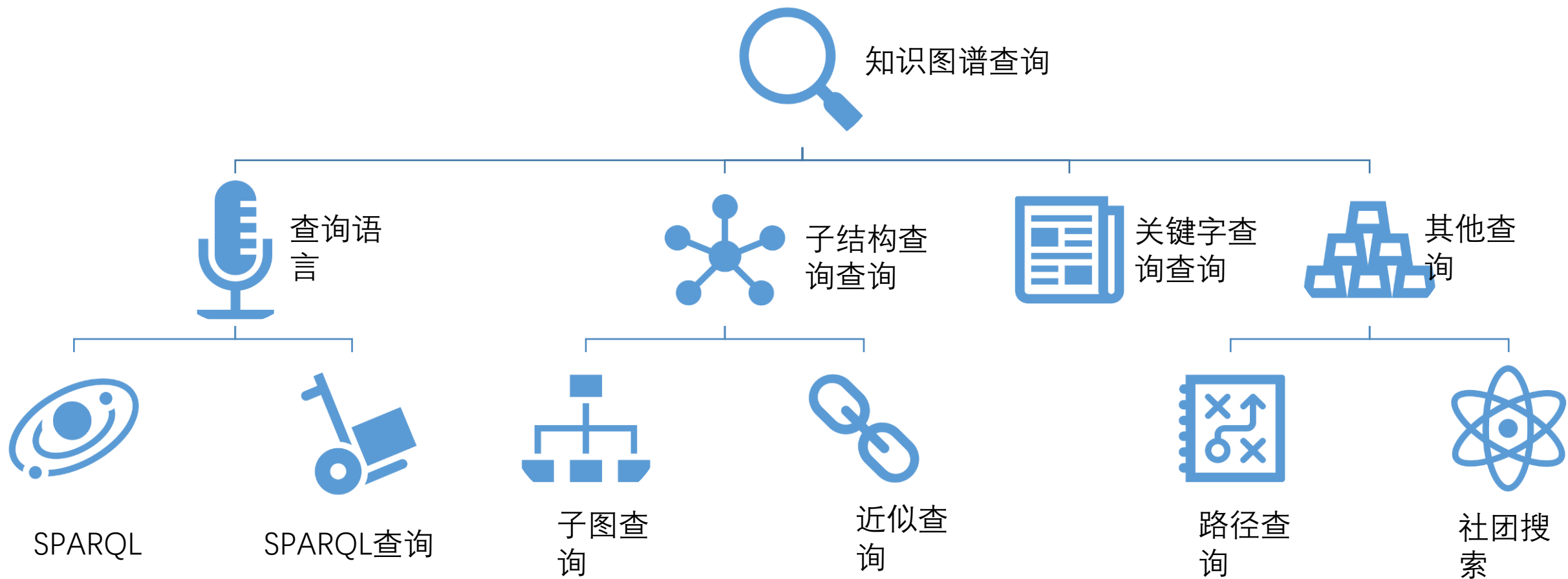
- 图

- 知识图谱 VS. 与一般的图
(general graph)



- 一般的图上的查询

知识图谱查询概述



查询语言： SPARQL

9.1.1 基本概念

- 语义网
 - 万维网之父蒂姆·伯纳斯-李(Tim Berners-Lee), 1998
 - 机器可以理解的 (machine understandable)
- 资源描述框架 (RDF, Resource Description Framework)
 - RDF三元组<主语(subject), 谓词(predicate), 宾语(object)>
- SPARQL
 - W3C 推荐的针对RDF数据的一种标准查询语言
 - 描述性的结构化查询语言

9.1.1 基本概念

- 三元组： (subject, predicate, object)
 - 空格分隔
 - “.”结束
 - subject: URI 或者空节点
 - predicate: URI
 - object: URI、空节点、或常量 (literal)
 - 常量: 字符串、数字、日期
- `<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .`



URI

9.1.1 基本概念

- 谓词-客体列表:

```
<http://example.org/sman> <http://xmlns.com/foaf/0.1/name> "Superman" ;  
                        <http://xmlns.com/foaf/0.1/Sex> "Man" .
```

```
<http://example.org/sman> <http://xmlns.com/foaf/0.1/name> " Superman" .  
                        <http://example.org/sman> <http://xmlns.com/foaf/0.1/Sex> "Man" .
```

- 对象列表:

- 多条三元组共享同样的主体和谓词

```
<http://example.org/sman> <http://xmlns.com/name> "Flashman" , "Superman" .
```

```
<http://example.org/sman> <http://xmlns.com/name> " Flashman" .  
<http://example.org/sman> <http://xmlns.com/name> "Superman" .
```

9.1.1 基本概念

- 前綴和IRI

Prefix	IRI
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
xsd:	http://www.w3.org/2001/XMLSchema#
somePrefix:	http://www.perceive.net/schemas/relationship/

PREFIX somePrefix: <<http://www.perceive.net/schemas/relationship/>> .

PREFIX expPrefix: <<http://example.org/>> .

expPrefix:Darkseid somePrefix:enemyOf expPrefix:Superman .

9.1.1 基本概念

- 注释：
 - 用“#”表示注释，即“#”字符后面该行的内容不被作为SPARQL查询的组成部分
- 查询变量：
 - 使用“?”或“\$”进行标记，但是“?”或“\$”并不是变量名的一部分
- 查询结果

x	y	z
“Catherine”	<http://example/a>	

9.1.2 简单查询

- SELECT子句
 - 识别查询的结果中将要返回的变量
- WHERE子句
 - 提供了需要在目标数据图上进行匹配的基本图模式
 - 数据:

```
<http://example.org/book1>      <http://purl.org/dc/title>      "Knowledge graph" .
```

- 查询: 书的名字

```
SELECT ?title  
WHERE  
{ <http://example.org/book1> <http://purl.org/dc/title> ?title . }
```

title

"Knowledge graph"

9.1.2 简单查询

- 创建数据-CONCAT函数
 - 把若干个字符串拼接起来，然后将拼接出来的值通过SELECT子句赋给一个返回变量

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:givenName "John" .  
_:a foaf:surname "Dora" .
```

```
PREFIX foaf: http://xmlns.com/foaf/0.1/  
  
SELECT ( CONCAT(?G, " ", ?S) AS ?name )  
  
WHERE { ?P foaf:givenName ?G ; foaf:surname ?S  
}
```

name

"John Dora"

9.1.3 查询约束

- FILTER函数

- 对于满足基本图模式的答案
- 如果FILTER表达式的布尔值为false，相应答案被过滤

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix : <http://example.org/book/> .  
@prefix ns: <http://example.org/ns#> .  
  
:book1 dc:title "SPARQL Tutorial" .  
:book1 ns:price 45 .  
:book2 dc:title "The Semantic Web" .  
:book2 ns:price 26 .
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>  
SELECT ?title  
WHERE { ?x dc:title ?title  
        FILTER regex(?title, "^SPARQL")  
}
```

限定字符串,正则表达式

title

"SPARQL Tutorial"

9.1.3 查询约束

- FILTER函数
 - 限定数值
 - 根据算术表达式进行约束

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .

:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 45 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 26
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: http://example.org/ns#

SELECT ?title ?price
WHERE { ?x ns:price ?price .
        FILTER (?price < 30.5)
        ?x dc:title ?title . }
```

title	price
"The Semantic Web"	23

9.1.4 可选匹配

- OPTIONAL函数

- 基本图模式需要严格匹配
- 可选匹配中的图模式
 - 如果能有子图与之匹配，则把相应的匹配中与变量对应的值返回
 - 否则，不影响基本图模式的匹配，涉及到的相关返回变量则为空
- 语法形式： pattern OPTIONAL { pattern }。

```
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a rdf:type     foaf:Person .
_:a foaf:name    "Catherine" .
_:a foaf:mbox    <mailto:catherine@example.com> .
_:a foaf:mbox    <mailto:catherine@work.example> .

_:b rdf:type     foaf:Person .
_:b foaf:name    "Bob" .
```


9.1.4 可选匹配

• OPTIONAL函数

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a rdf:type foaf:Person .
_:a foaf:name "Catherine" .
_:a foaf:mbox <mailto:catherine@example.com> .
_:a foaf:mbox <mailto:catherine@work.example> .

_:b rdf:type foaf:Person .
_:b foaf:name "Bob" .

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name ?mbox  
WHERE { ?x foaf:name ?name .  
        OPTIONAL { ?x foaf:mbox ?mbox } }
```

name	mbox
"Catherine"	<mailto:alice@example.com>
"Catherine"	<mailto:alice@work.example>
"Bob"	

9.1.4 可选匹配

- OPTIONAL函数

```
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .  
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
_:a rdf:type      foaf:Person .  
_:a foaf:name     "Catherine" .  
_:a foaf:mbox     <mailto:catherine@example.com> .  
_:a foaf:mbox     <mailto:catherine@work.example> .
```

```
_:b rdf:type      foaf:Person .  
_:b foaf:name     "Bob" .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name ?mbox ?hpage  
WHERE { ?x foaf:name ?name .  
        OPTIONAL { ?x foaf:mbox ?mbox } .  
        OPTIONAL { ?x foaf:homepage ?hpage }  
}
```

包含多条三元组模式

支持嵌套

9.1.5 联合匹配

- UNION函数

- 把多个基本图查询模式的匹配结果融合到一起的方法
- 不同的基本图模式用“UNION”进行连接，每组基本图模式用“{ }”进行界定

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { { ?book dc10:author ?author }
        UNION
        { ?book dc11:author ?author } }
```

- 其它高级运算操作

- NOT EXISTS, EXISTS, MINUS, VALUES和属性路径等，此外还支持聚集操作，常见的语法有GROUP BY, COUNT, SUM, MIN, MAX, AVG, SAMPLE和 GROUP_CONCAT。
- <https://www.w3.org/TR/sparql11-query/#negation>

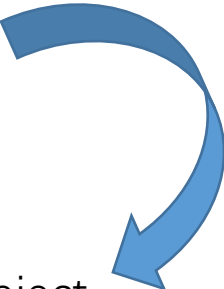
SPARQL查询

9.2 SPARQL查询

- 基于关系表

```
SELECT ?a
WHERE {
  ?a <bornIn> ?city . ?a <graduateFrom> Stanford_University .
  ?p <author> ?a . ?p <publishedYear> " 2005 " .
  FILTER( regex( str(?city), "York" ) )
}
```

Subject	Property	Object
y:David_Blanco	hasName	"David Blanco"
y:David_Blanco	bornOnDate	"1964-02-12"
y:David_Blanco	bornIn	y:New_York
y:David_Blanco	gender	"Male"
y:New_York	state	"New York"
y:New_York	country	y:United_States
y:United_States	hasName	"United States"
y:David_Blanco	graduateFrom	y:Stanford_University
y:p1	title	"knowledge graph"
y:p1	author	y:David_Blanco
y:p1	publishedYear	"2005"
y:p2	title	"data mining"
y:p2	author	y:Michael_John
y:p2	author	y:David_Blanco



```
SELECT T1.subject
FROM T as T1, T as T2, T as T3, T as T4
WHERE T1.property="bornIn"
AND T2.property="gradudateFrom"
AND T3.property="author"
AND T4.property="pubishedYear"
AND T1.subject=T2.subject
AND T2.object="Stanford_University"
AND T3.object=T1.subject
AND T3.subject=T4.subject
AND T1.object LIKE '%York%'
```

9.2 SPARQL查询

- 基于关系表
 - 属性表

Subject	Property	Object
y:David_Blanco	hasName	"David Blanco"
y:David_Blanco	bornOnDate	"1964-02-12"
y:David_Blanco	bornIn	y:New_York
y:David_Blanco	gender	"Male"
y:New_York	state	"New York"
...

Subject	hasName	gender	bornIn
y:David_Blanco	"David Blanco"	"Male"	New York
y:Christian_Kaabi	"Christian Kaabi"	"Female"	

Subject	Title	publishedYear
y:p1	"data mining"	"2005"
y:p2	"knowledge graph"	

减少连接操作

空值？多值？

9.2 SPARQL查询

- 基于关系表
 - 属性表
 - 二分表（垂直划分）

Subject	Property	Object
y:David_Blanco	hasName	"David Blanco"
y:David_Blanco	bornOnDate	"1964-02-12"
y:David_Blanco	bornIn	y:New_York
y:David_Blanco	gender	"Male"
y:New_York	state	"New York"
...

hasName

Subject	Object
y:David_Blanco	"David Blanco"
y:Christian_Kaabi	"Christian Kaabi"

title

Subject	Object
y:p1	"data mining"
y:p2	"knowledge graph"

解决了空值，多值

主体-客体的连接

9.2 SPARQL查询

- 基于关系表
 - 属性表
 - 二分表（垂直划分）
 - 全索引
 - SPO, SOP, PSO, POS, OPS, OSP

连接操作 → 归并操作

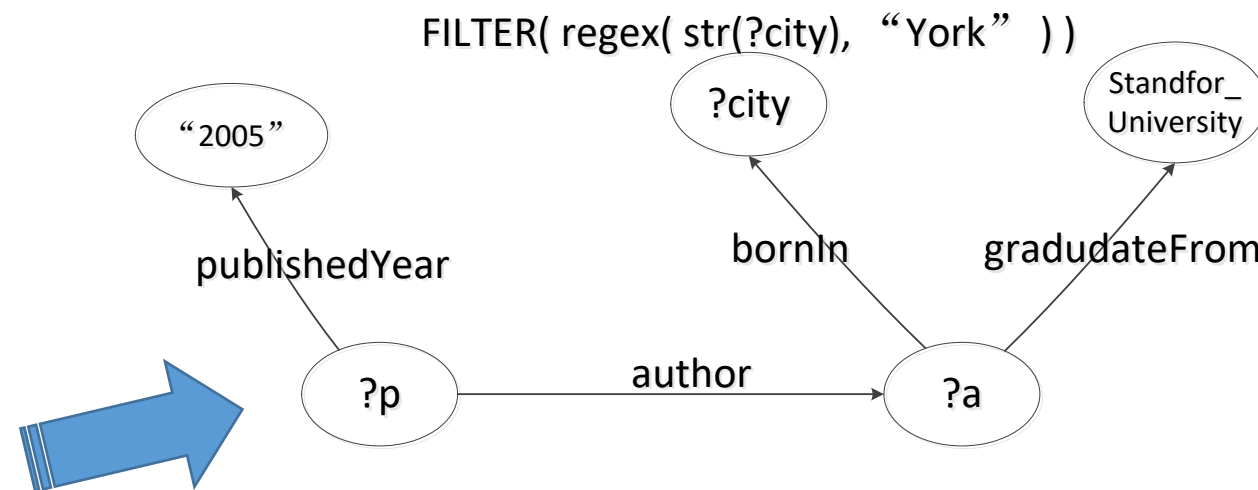
空间开销 巨大

9.2 SPARQL查询

- 基于关系表
 - 属性表
 - 二分表（垂直划分）
 - 全索引
 - SPO, SOP, PSO, POS, OPS, OSP

- 基于子图查询

```
SELECT ?a
WHERE {
  ?a <bornIn> ?city . ?a <graduateFrom> Stanford_University .
  ?p <author> ?a .   ?p <publishedYear> " 2005 " .
  FILTER( regex( str(?city), "York" ) )
}
```



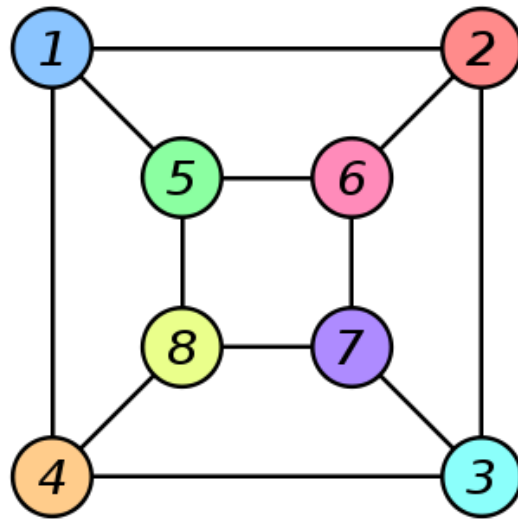
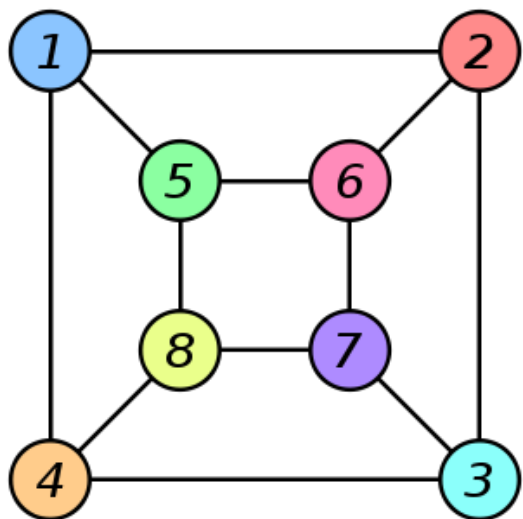
子结构查询

9.3.1 子图匹配

- 图同构^[1]

- 对于给定的两个图 $g=\{V, E, \Sigma, L\}$ 和 $g'=\{V', E', \Sigma', L'\}$, 当且仅当存在一个函数 f 满足如下条件时:

- (1) 对于 V 中任意一个顶点 v , 满足 $L(v)=L'(f(v))$, 并且 $f(v) \in V'$;
- (2) 对于 V 中的任意两个顶点 v_1 和 v_2 , 如果 v_1 和 v_2 之间有一条边 $e(v_1, v_2) \in E$, 在 E' 中都有一条边与之对应 $e(f(v_1), f(v_2)) \in E'$ 。

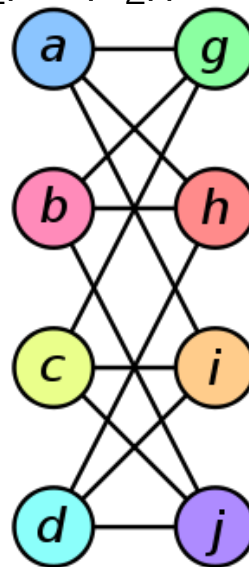
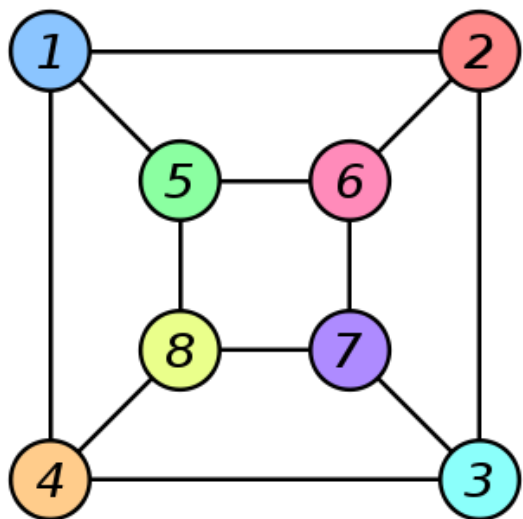


9.3.1 子图匹配

- 图同构

- 对于给定的两个图 $g=\{V, E, \Sigma, L\}$ 和 $g'=\{V', E', \Sigma', L'\}$, 当且仅当存在一个函数 f 满足如下条件时:

- (1) 对于 V 中任意一个顶点 v , 满足 $L(v)=L'(f(v))$, 并且 $f(v) \in V'$;
- (2) 对于 V 中的任意两个顶点 v_1 和 v_2 , 如果 v_1 和 v_2 之间有一条边 $e(v_1, v_2) \in E$, 在 E' 中都是一条边与之对应 $e(f(v_1), f(v_2)) \in E'$ 。



$$f(a) = 1$$

$$f(b) = 6$$

$$f(c) = 8$$

$$f(d) = 3$$

$$f(g) = 5$$

$$f(h) = 2$$

$$f(i) = 4$$

$$f(j) = 7$$

9.3.1 子图匹配

- 图同构

- 对于给定的两个图 $g=\{V, E, \Sigma, L\}$ 和 $g'=\{V', E', \Sigma', L'\}$, 当且仅当存在一个函数 f 满足如下条件时:

- (1) 对于 V 中任意一个顶点 v , 满足 $L(v)=L'(f(v))$, 并且 $f(v)\in V'$;
- (2) 对于 V 中的任意两个顶点 v_1 和 v_2 , 如果 v_1 和 v_2 之间有一条边 $e(v_1, v_2)\in E$, 在 E' 中也都有一条边与之对应 $e(f(v_1), f(v_2))\in E'$ 。

- 子图同构

- 如果图 G 中至少存在一个子图 g 使得 q 同构于 g , 则认为图 q 子图同构于图 G
- Ullmann算法^[5], VF2算法^[6]

9.3.1 子图skyline

- 子图支配 (dominate)
 - 给定两个图 g_1 和 g_2 , 我们认为 g_1 支配(dominate) g_2 , 需要满足如下条件:
 - (1) 如果不考虑 g_1 和 g_2 中数值节点, g_1 和 g_2 是图同构的;
 - (2) 对于 g_1 中的每个数值节点 v_i 都支配 $f(v_i)$, 即 v_i 的值不比 $f(v_i)$ 的值大;
 - (3) g_1 中至少存在一个节点 v_j 比 $f(v_j)$ 小。
 - 其中 $f(\cdot)$ 是 g_1 和 g_2 的之间的映射函数。
- 子图skyline^[2]
 - 如果不考虑数值型的节点 G 中的子图 g 与查询图 q 同构,
 - 并且关于 q 中的数值型节点图 g 不被 G 的任何其它子图支配,

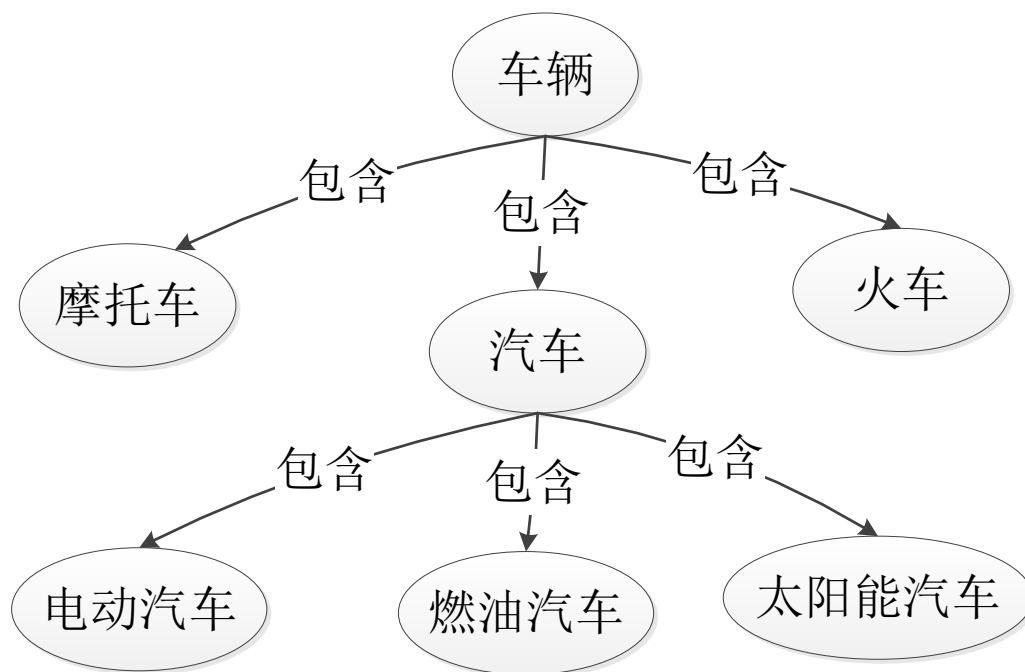
9.3.2 近似子结构查询

- 子图匹配
 - 十分严格 → 没有匹配
 - 知识图谱：实体，以及实体间关系 → 语义信息
- 基于本体的近似查询
- 融合结构和语义的近似子结构查询

9.3.2 基于本体的近似查询

- 本体

- 知识图谱中的本体描述了知识图谱中概念之间的关联
- 概念、实体
- 语义关系



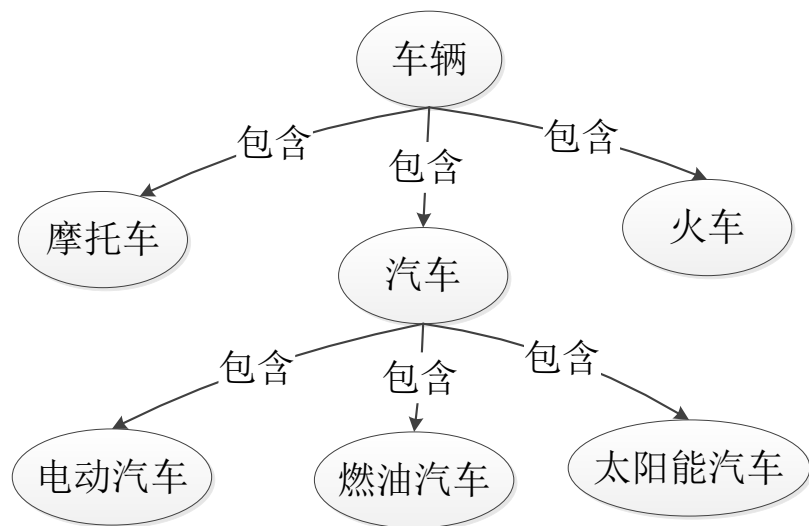
9.3.2 基于本体的近似查询

- 本体节点之间的相似度
 - 最近公共祖先节点的深度

- 电动汽车,燃油汽车
- 太阳能汽车,火车

- $\text{sim}(\text{电动汽车}, \text{燃油汽车}) > \text{sim}(\text{太阳能汽车}, \text{火车})$

- 本体图中的距离^[7]
 - 越远, 相似度越小



9.3.2 基于本体的近似查询

- 近似匹配

- 对于查询图 $q=\{V, E, \Sigma, L\}$ 和目标图 G 的某个子图 $g=\{V', E', \Sigma', L'\}$, 当且仅当存在一个函数 f 满足如下条件时, 我们认为 g 是 q 的一个匹配
 - (1) 对于 V 中任意一个顶点 v , 满足 $\text{sim}(L(v), L'(f(v))) \geq \theta$, 并且 $f(v) \in V'$;
 - (2) 对于 V 中的任意两个顶点 v_1 和 v_2 , 如果 v_1 和 v_2 之间有一条边 $e(v_1, v_2) \in E$, 在 E' 中也都有一条边与之对应 $e(f(v_1), f(v_2)) \in E'$ 。

- 匹配的质量

- 所有的对应的顶点对之间的相似度求和

$$\text{sim}(q, g) = \sum \text{sim}(L(v), L'(f(v))), \quad v \in V$$

9.3.2 基于本体的近似查询

- 匹配的质量 $\text{sim}(q, g) = \sum \text{sim}(L(v), L'(f(v))), \quad v \in V$

优势

Top-k结果

考虑节点语义

缺点

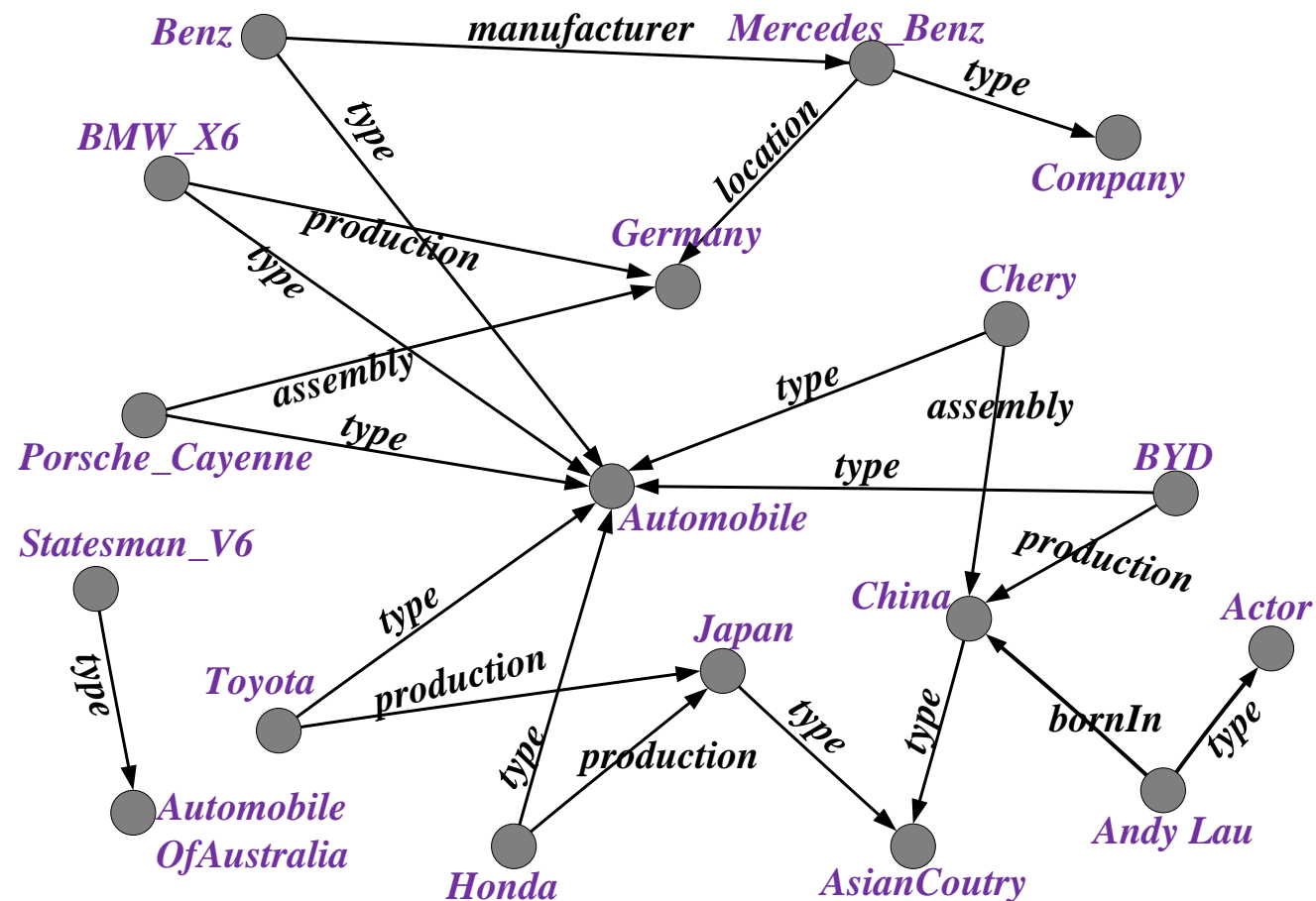
计算复杂度高
($\theta = 1$, 精确匹配)

θ , top-k函数
难以指定

没有考虑结构相似

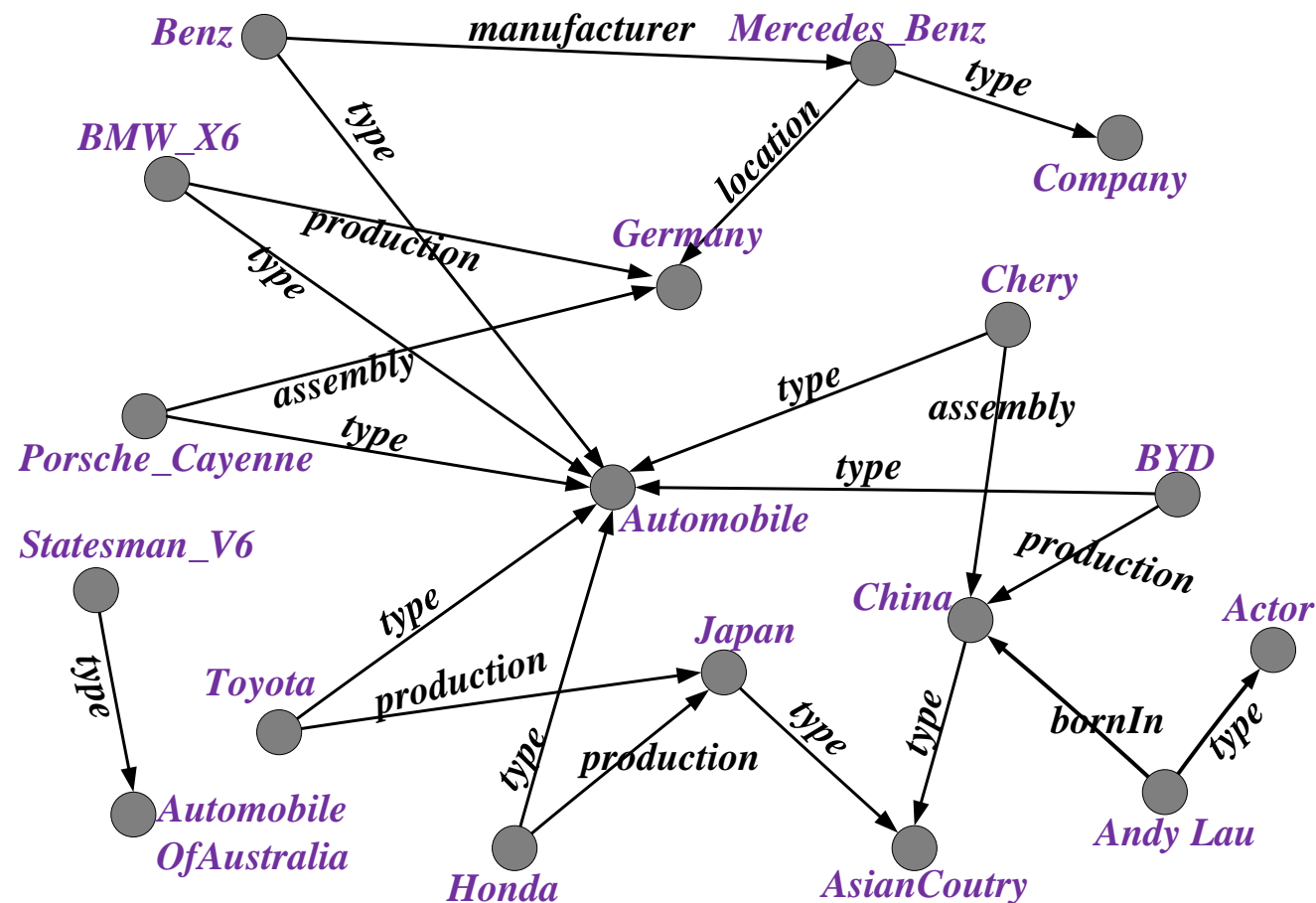
9.3.2 融合结构和语义的近似子结构查询

- 基于本体的近似查询
 - 节点之间的语义相似
- 结构上的语义相似^[4]
 - 德国生产的汽车



9.3.2 融合结构和语义的近似子结构查询

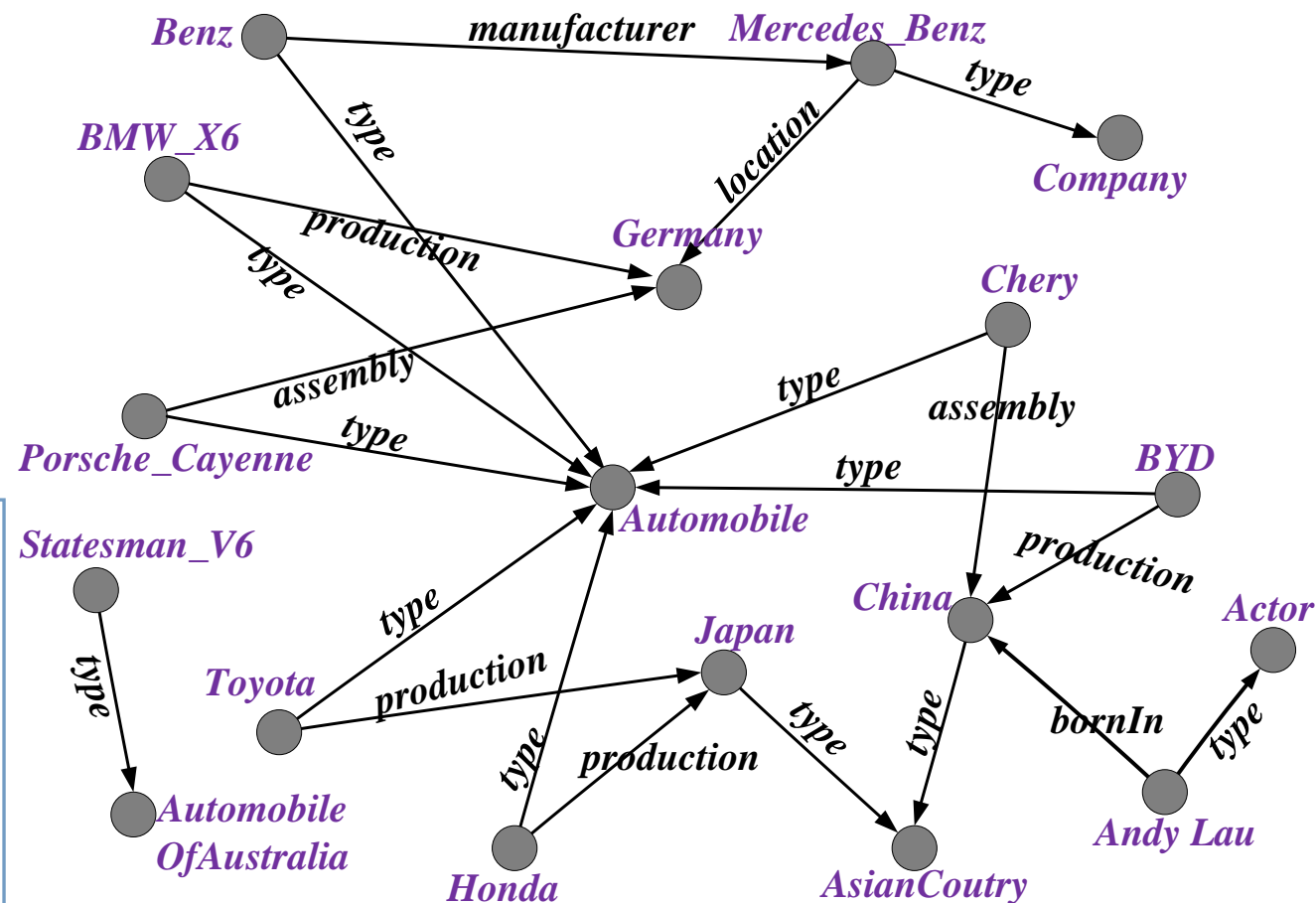
- 基于本体的近似查询
 - 节点之间的语义相似
- 结构上的语义相似^[4]
 - 德国生产的汽车



9.3.2 融合结构和语义的近似子结构查询

- 基于本体的近似查询
 - 节点之间的语义相似
- 结构上的语义相似^[4]
 - 德国生产的汽车

```
SELECT ?x WHERE {  
  {?x <type> Automobile . ?x <production> Germany . }  
  UNION  
  {?x <type> Automobile . ?x <assembly> Germany . }  
  UNION  
  {?x <type> Automobile . ?x <manufacturer> ?y .  
   ?y <location> Germany .}}
```



9.3.2 语义图编辑距离

- 图编辑距离

- 增加顶点、删除顶点、增加边、删除边、替换顶点标签、替换边上标签
- 把q转换成g所需要的最小的图编辑操作数
- 缺乏对语义支持

- 语义图编辑距离

- 语义顶点增加

- 根据所增加的顶点的类型计算出其增加操作的代价（不再是1）
- 本体图中的两个概念s和t
 - 根据向上公共主题距离（upward cotopic distance）来计算

概念s在本体图中的祖先概念

$$dist(s, t) = 1 - \frac{|S(s, O) \cap S(t, O)|}{|S(s, O) \cup S(t, O)|}$$

9.3.2 语义图编辑距离

- 语义图编辑距离

- 语义顶点增加

- 语义顶点删除

- 删除一个类型为 t 的顶点，其编辑距离是 $dist(s, t)$ ， s 为本体图中的根节点

- 语义顶点替换

- 语义边增加

- 增加一条标签（谓词）为 r 的边，编辑代价是 r 和谓词本体图中的根节点 r_0 之间的语义距离，即 $dist(r, r_0)$

- 语义边删除

- $dist(r, r_0)$

- 语义边替换

- $dist(r_1, r_2)$

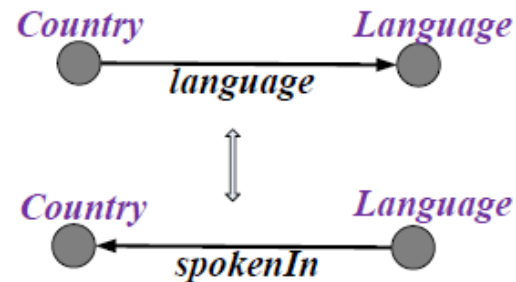
$$dist(s, t) = 1 - \frac{|S(s, O) \cap S(t, O)|}{|S(s, O) \cup S(t, O)|}$$

9.3.2 语义图编辑距离

- 语义图编辑距离^[4]

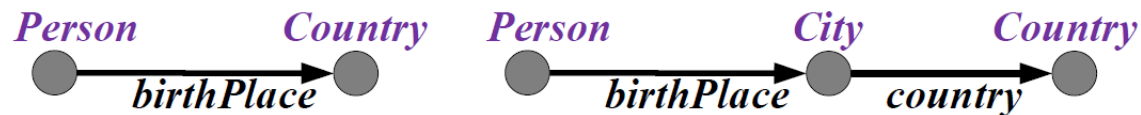
- 语义边重定向

- $(v_1, r_1, v_2) \rightarrow (v_2, r_2, v_1)$
 - 更改边的方向,
 - 用谓词 r_2 替换该条边上的原始谓词 r_1
 - 替换 r_1 的 r_2 需要有一定的限制, 即符合语义对等的要求 (语义挖掘和计算)
 - $(\text{person}_1, \text{influenced}, \text{person}_2) \rightarrow (\text{person}_2, \text{influencedBy}, \text{person}_1)$



- 语义路径替换

- 用一条边 e 替换一条路径 p 或者使用一条路径替换一条边。



- 语义星型替换

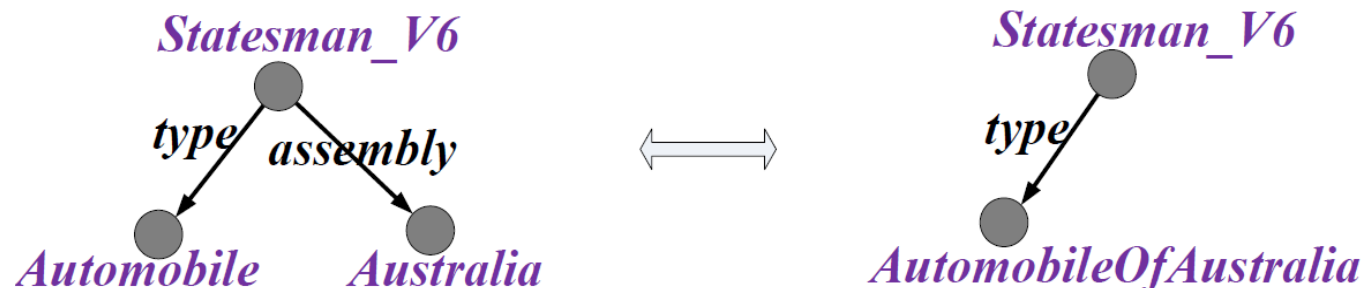
- 用一条边 $(v, type, t)$ 替换一个星状结构或用一个星状结构替换一条边。

9.3.2 语义图编辑距离

- 语义图编辑距离^[4]

- 语义星型替换

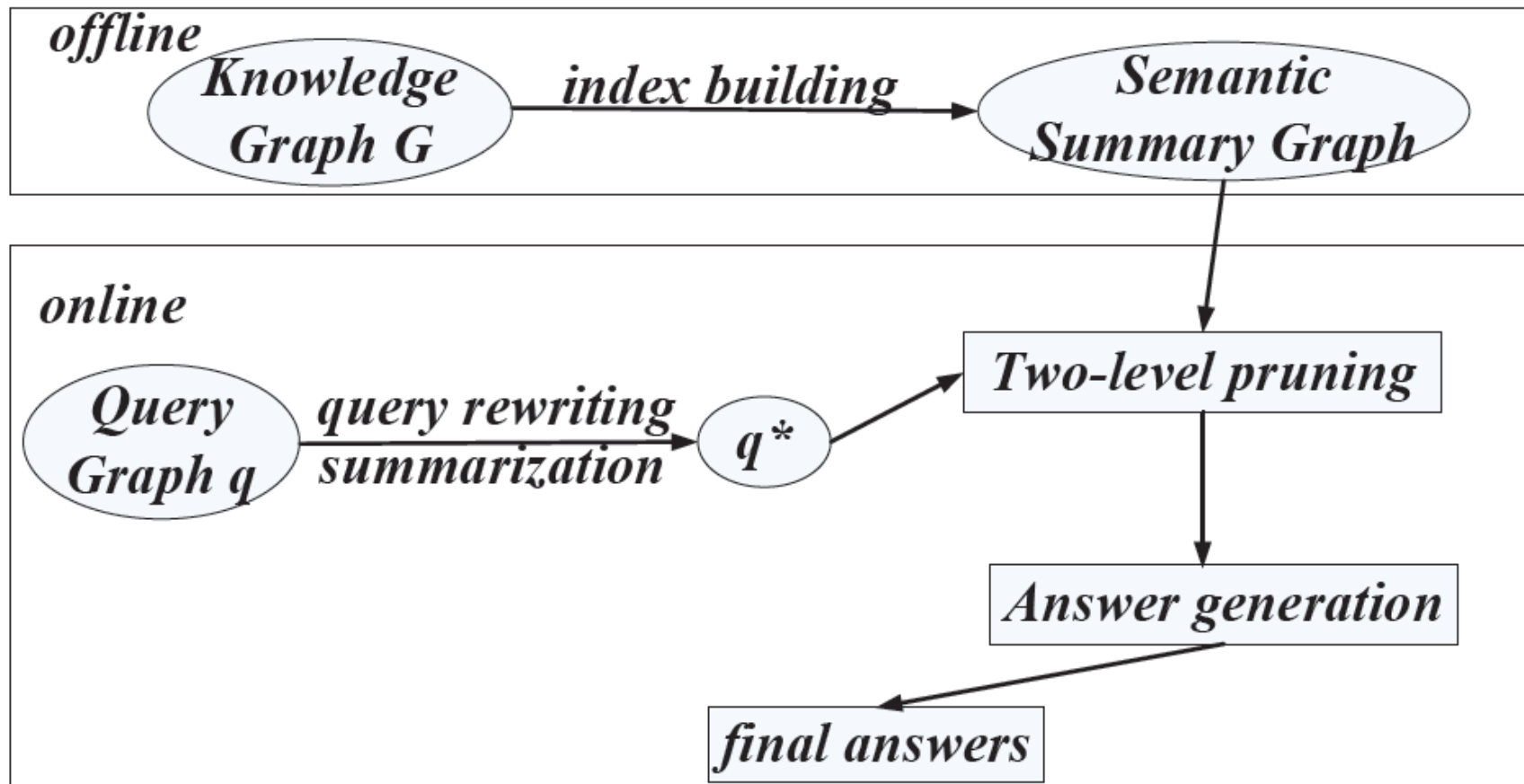
- 用一条边 $(v, type, t)$ 替换一个星状结构或用一个星状结构替换一条边。
 - 星型结构中一般包含一个类型 t' ，并且类型 t 是 t' 的子孙类型， v 的邻居组合起来描述了类型 t 的信息



后三种语义编辑操作代价均是0

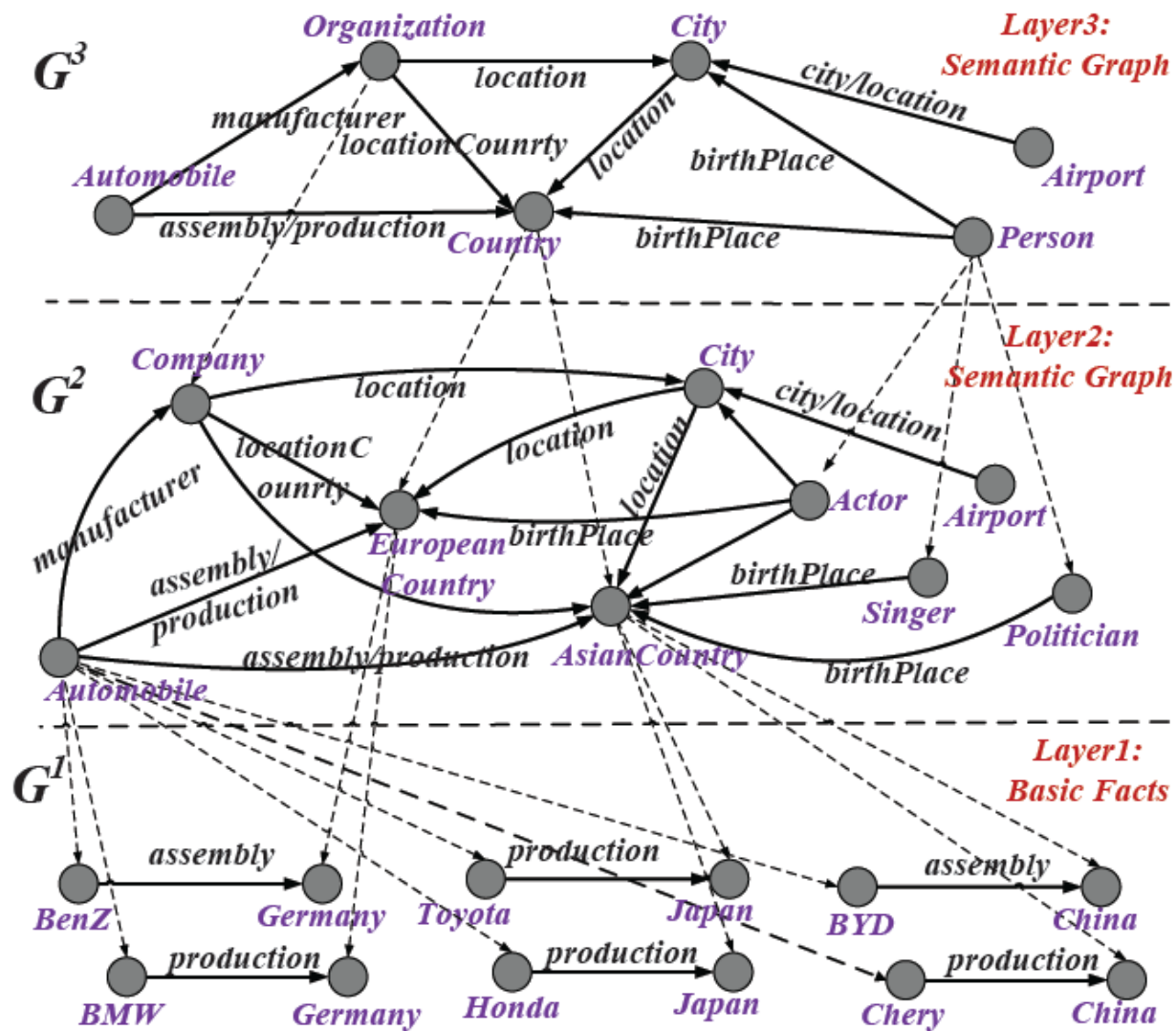
语义图编辑距离: g_1 转换为 g_2 所需要的最小的语义图编辑操作的代价

9.3.2 语义图编辑距离



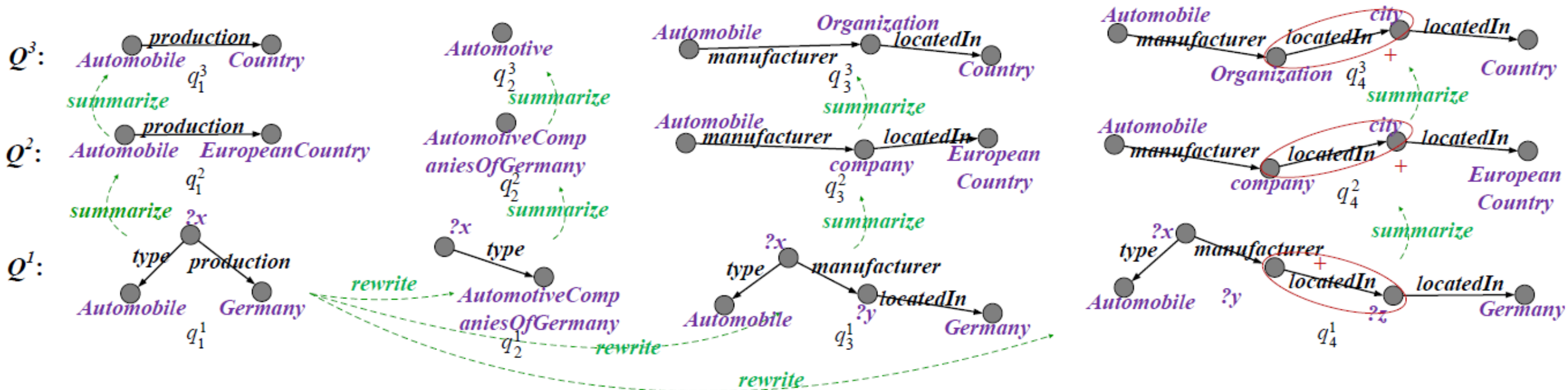
9.3.2 语义图编辑距离

- 索引结构
 - 语义抽象图^[4]



9.3.2 语义图编辑距离

- 查询重写^[4]



9.3.3 Top-k查询

- 一般top-k查询

景点	推荐指数1
九寨沟	10
大理	8
西湖	7
黄山	6

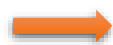
景点	推荐指数2
大理	9
西湖	7
九寨沟	6
黄山	6

- $R(\text{九寨沟}) = (10+6)/2=8$
 $R(\text{大理}) = (8+9)/2=8.5$
 $R(\text{西湖}) = (7+7)/2=7$
 $R(\text{黄山}) = (6+6)/2=6$

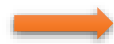
暴力计算-低效

9.3.3 Top-k查询

- 一般top-k查询
- Fagin算法

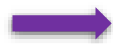


景点	推荐指数1
九寨沟	10
大理	8
西湖	7
黄山	6

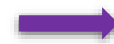


景点	推荐指数2
大理	9
西湖	7
九寨沟	6
黄山	6

推荐指数1	推荐指数2
九寨沟 (10)	
	大理 (9)



推荐指数1	推荐指数2
九寨沟 (10)	
	大理 (9)
大理 (8)	西湖 (7)



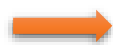
推荐指数1	推荐指数2
九寨沟 (10)	
	大理 (9)
大理 (8)	
	九寨沟 (6)

9.3.3 Top-k查询

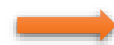
- 一般top-k查询

- Threshold

算法



景点	推荐指数1
九寨沟	10
大理	8
西湖	7
黄山	6



景点	推荐指数2
大理	9
西湖	7
九寨沟	6
黄山	6

推荐指数1	推荐指数2	AVG
九寨沟 (10)	大理 (9)	9.5

推荐指数1	推荐指数2	AVG
九寨沟 (10+6)	大理 (9+8)	9.5

推荐指数1	推荐指数2	AVG
九寨沟 (10+6)	大理 (9+8)	9.5
大理 (8)	西湖 (7)	7.5

9.3.3 Top-k查询

- 目标: 返回和理想值最接近的前k个匹配或答案

先找到候选结果，然后排序，最后再选择top-k



9.3.3 Top-k查询

- 目标: 返回和理想值最接近的前k个匹配或答案

- 查询框架

- 先过滤后验证

- 过滤阶段

- 先找到k个任意匹配放进答案列表（相似度：高→低）
 - 计算候选匹配的上界或下界

先找到候选结果，然后排序，最后再选择top-k



LowerBound($\text{sim}(q, A_p)$) $> A_k$, 则进入答案
UpperBound($\text{sim}(q, A_p)$) $< A_k$, 则停止搜索

关键字查询

9.4 关键字查询

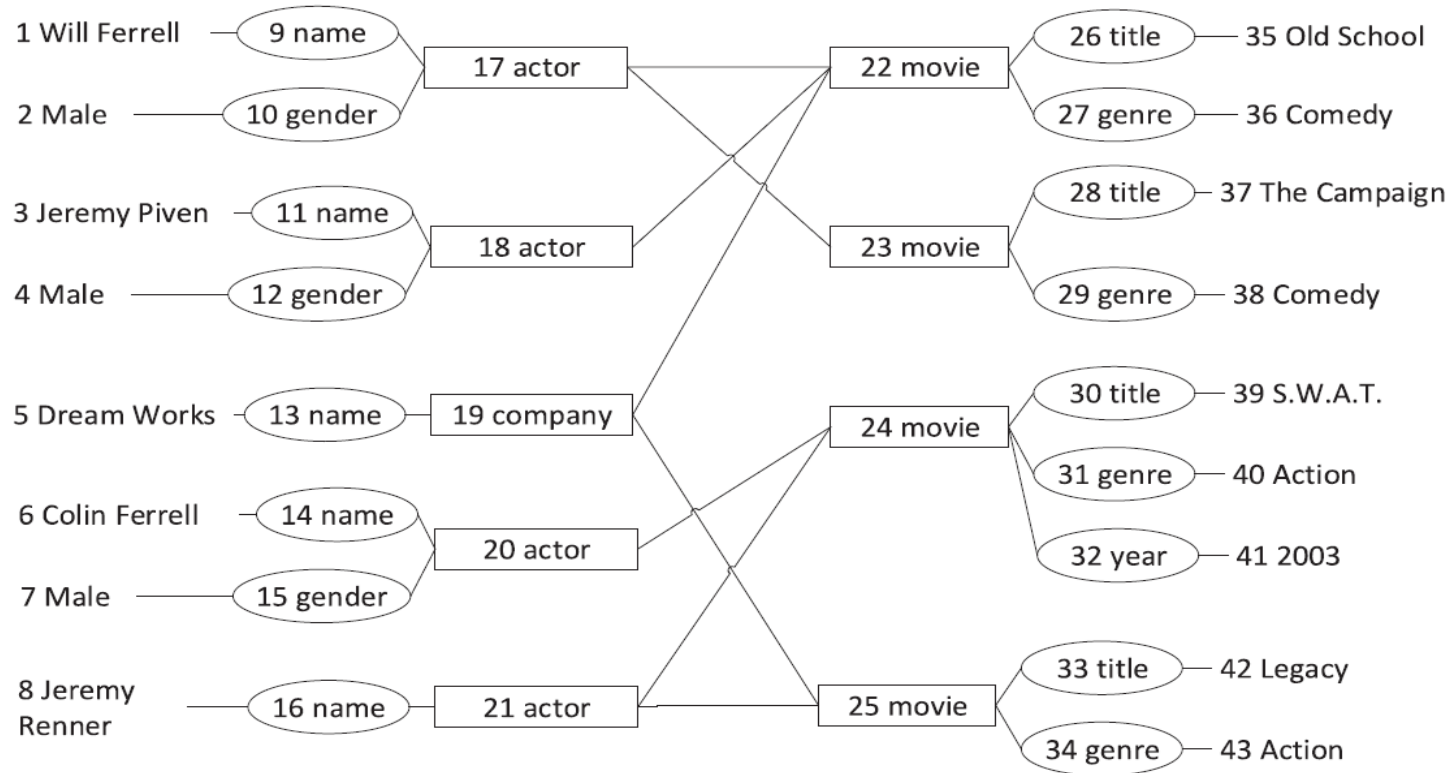
- 一般图数据上的关键字查询
 - 产生一个子树或子图,
 - 这个子树或子图包含查询中的所有关键字同时是最小的
 - 最小斯坦纳树 (Steiner Tree)
 - 使得给定的所有顶点集合连通, 并且边权总和最小的生成树

关注效率

改进答案的质量

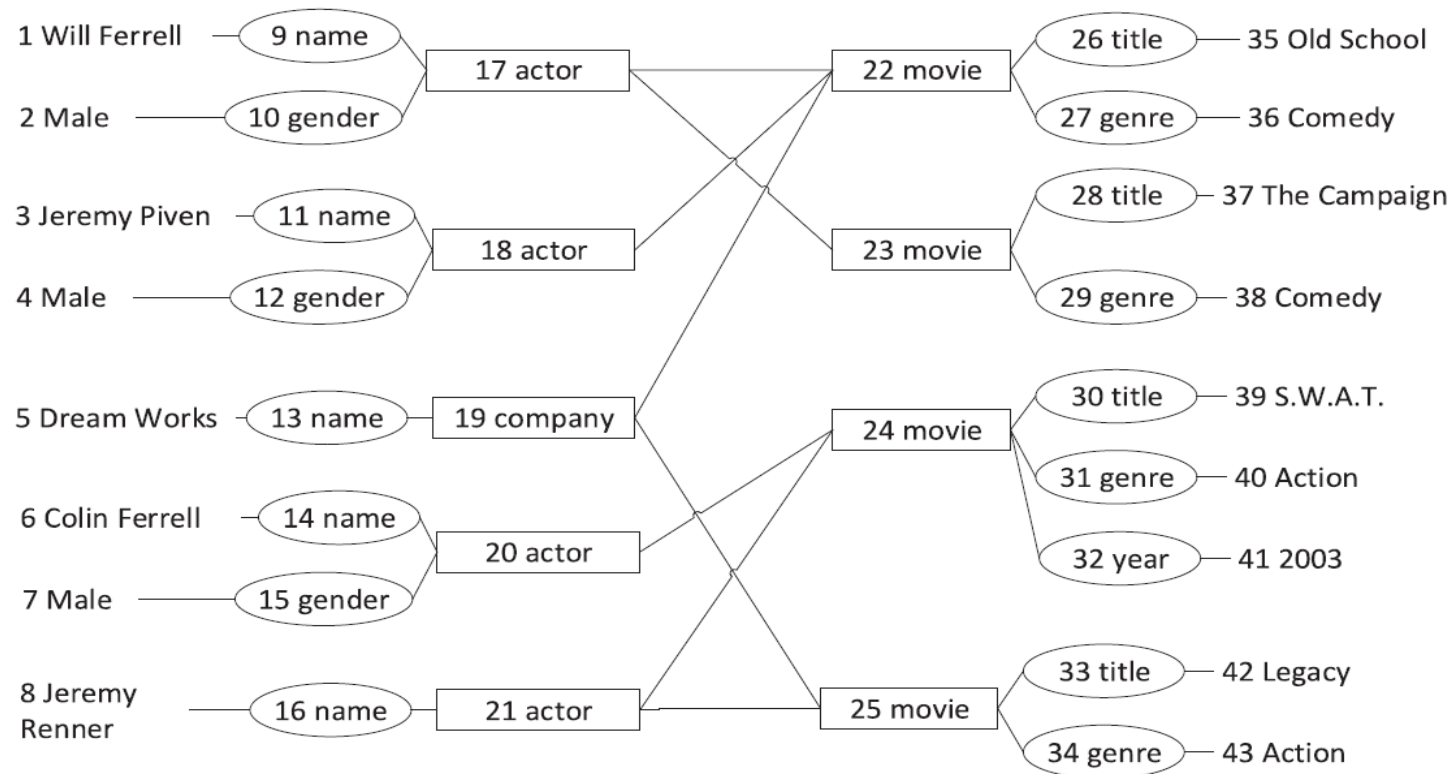
9.4 关键字查询

- “comedy, actor”
- 演过喜剧电影的演员^[3]



9.4 关键字查询

- “comedy, actor”
- 演过喜剧电影的演员^[3]



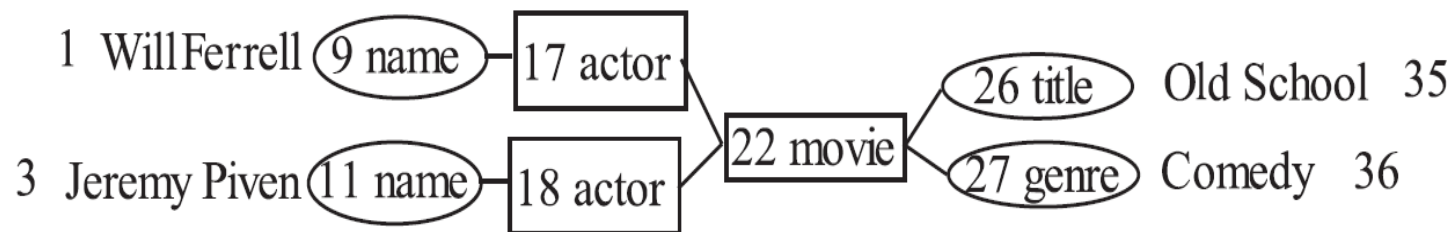
(a) 1 Will Ferrell (9 name) — 17 actor — 22 movie (26 title Old School 35, 27 genre Comedy 36)

(b) 1 Will Ferrell (9 name) — 17 actor — 23 movie (28 title The Campaign 37, 29 genre Comedy 38)

9.4 关键字查询

- 包含所有关键字
 - 但不一定最小

无法选出用户最感兴趣的前k个演员



- 体现语义
 - 核心意图词、修饰词
- 提升语义表达能力
 - 执行多次查询
 - 对结果分类
 - 结构、内容信息

其他查询

9.5.1 路径查询

- 一般的图
 - 最短路径查询
- 基于限制的路径查询
 - 白名单标签限制路径
 - 路径上的边标签集合是预先指定的白名单标签集合的一个子集
 - 黑名单限制路径：
 - 路径上的边标签不能出现在预先指定的黑名单标签集合中
- 基于白名单标签限制的路径查询
 - 给定一个起始顶点 s 和目标顶点 t ，以及白名单标签集合 T ，其任务是返回从 s 到 t 的最短路径，并且路径上的边标签是 T 的子集

9.5.1 路径查询

- 基于黑名单标签限制的路径查询
 - 指定了黑名单标签集合S
 - 转化成白名单限制的路径查询，其中 $T=R \setminus S$
- 基于标记的方法
 - 提前选取一些点，然后计算所有点到这些点的最短距离
$$dist'(s, t) = \min_{l \in L} \{dist(s, l) + dist(l, t)\}$$
- 基于白/黑名单标签限制的路径查询
 - 提前检查这些预先存储的路径信息是否满足限制，然后进行筛选
 - 对边标签建立索引

9.5.1 路径查询

- 基于元路径（meta path）的查询
 - 对路径上的**顶点类型**进行限制
- 元路径: 是指由一系列指定的**关系**以及**实体类型**组成的路径
 - 会议-论文-关键字-论文-会议
 - 相同关键字（或主题）的会议
- 元路径自动发现:
 - 把给定的两个节点之间所有可能的元路径,
 - 选取相关程度最高的元路径

可解释性强

9.5.1 路径查询

- 元路径自动发现:

- 前向阶梯路径生成算法

- (1) 基于给定的相似节点先自动产生一些元路径
 - (2) 把这些元路径作为查询条件发现更多的相似节点
 - (3) 用户从中选取那些确实相似的结果

- 基于树结构进行扩展的方法

- 每个节点存储两部分的信息:
 - (1) 该节点或新发现的路径的重要程度SC
 - (2) 相似实体对以及它们之间当前的路径
 - 根节点包含所有的种子节点与其自身组成的节点对
 - 从根节点开始每次选取重要的元路径进行扩展

9.5.2 社团搜索

- 社团
 - 一组内部互相紧密联系的顶点，并且它们与社团之外的顶点联系相对松散
 - 社团密度：
 - 该子结构的平均顶点度数
 - 限定每个点的度数最小值
- 社团搜索应用
 - 团队推荐
 - 风险控制
 - 犯罪团伙发现

9.5.2 社团搜索

- 社团发现^[8,9]
 - 给一个图G和社团度量标准，找出其中所有的社团
- 社团查询^[10,11,12]
 - 个性化社团发现
 - 在线阶段给定一个查询顶点，目标是找到包含该顶点的社团

传统社团搜索
只考虑结构特征

9.5.2 社团搜索

- 基于属性的社团搜索^[13]
 - 在结构上社团内部的顶点联系非常紧密
 - 在语义上这些点彼此高度相关
 - 比如某个社团中的人都对音乐感兴趣或职业跟音乐相关，则可以清楚的了解这个社团的类型
- 属性的限制
 - 限定社团里的顶点的标签集合必须为某个集合的子集
 - 或社团中每个顶点都共享若干关键词

9.5.2 社团搜索

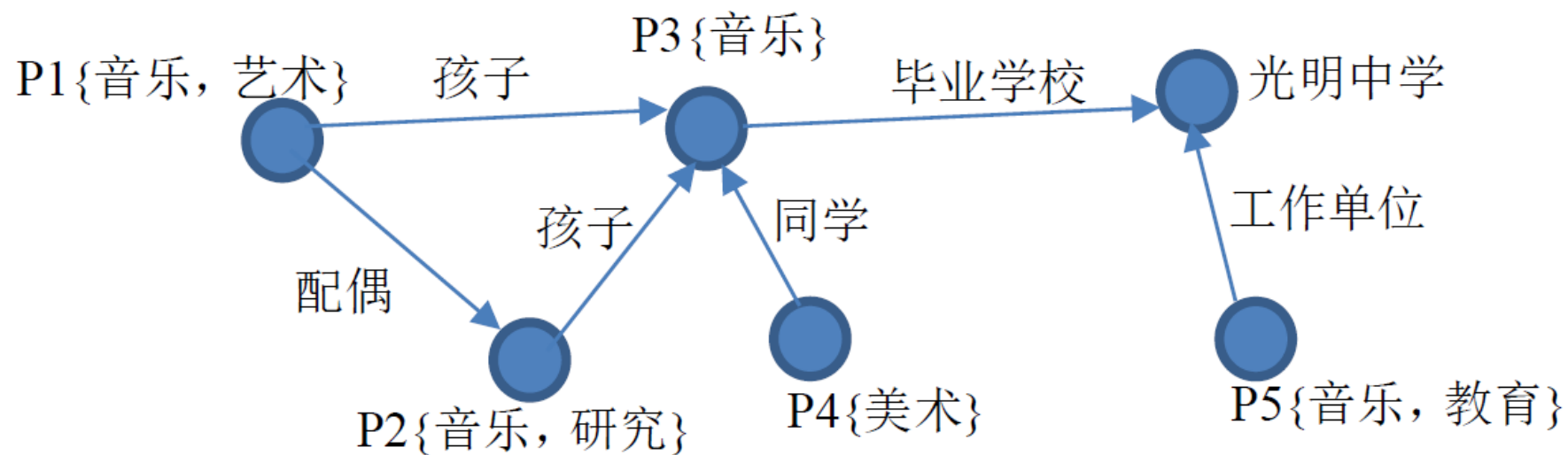
- 基于属性的社团搜索^[13]: $G(V,E)$, $k, q \in V$, key words S , 返回子图 G_q

- a) 连通性: G_q 连通的, 且包含 q
- b) 结构紧凑: 所有顶点 $v \in G_q$, $d(v) \geq k$
- c) 语义紧凑: G_q 中所有点的标签和 S 的交集最大

$$\bigcap_{v \in G_q} (W(v) \cap S)$$

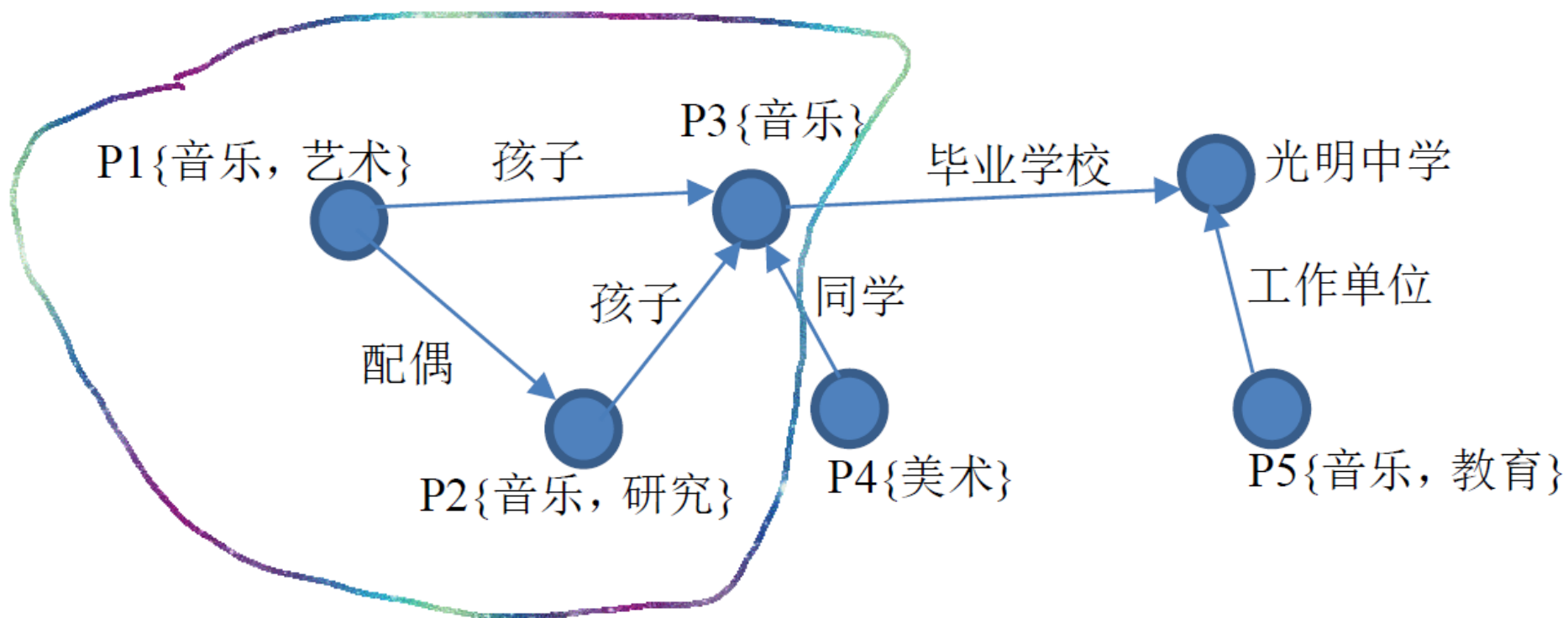
9.5.2 社团搜索

- 基于属性的社团搜索



9.5.2 社团搜索

- 基于属性的社团搜索



9.5.2 社团搜索

- 简单的方法

- (1) 枚举所有可能的关键词组合: $S_1, S_2, \dots, S_t, t = 2^{|S|} - 1$
- (2) 返回满足结构约束的子图
- $O(2^{|W|})$

- 基于索引的方法

- 把顶点上相关的关键词有效的组织起来
 - 层次的树形结构CL-tree

基本性质：对于一个关键词集合 S ，如果某个社团的每一个顶点都包含 S ，那么对于集合 S 的每个子集 S' ，一定存在一个满足 S' 约束的社团，即该社团中的每个顶点都包含 S'

9.5.2 社团搜索

- 如果基于 S' 不能找到一个 G_q , 则任何 S' 的超集也不用检查
- Size-1, Size-2, ...
 - 验证
 - Each S' in Size- c (初始时候 $c=1$), 当 $G_q(S')$ 存在, S' 标为合法
 - 候选生成
 - 对于仅有一个关键字不同的两个合法的 S_1 和 S_2 , 合并扩展成Size- $(c+1)$
 - $(S_1 \cup S_2)$ 的所有子集都合法

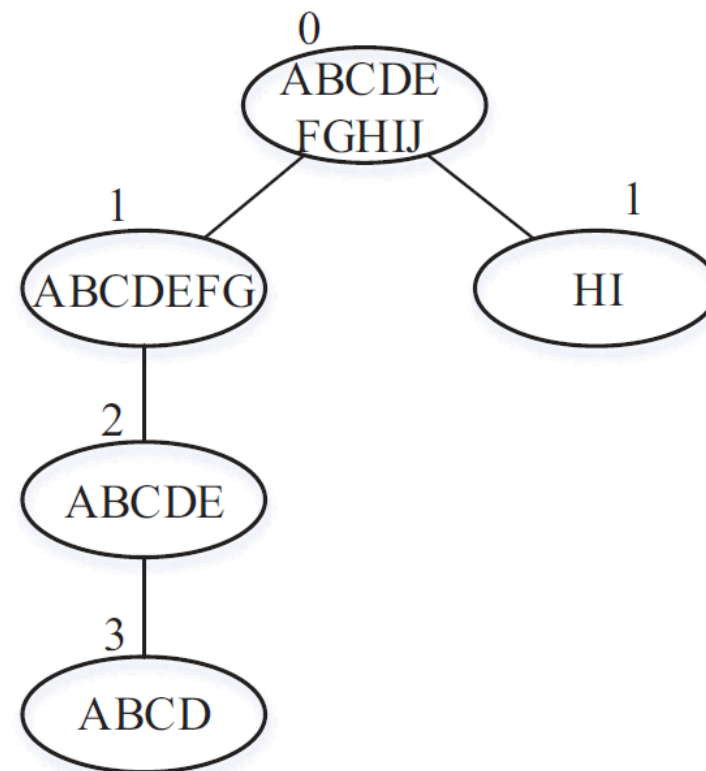
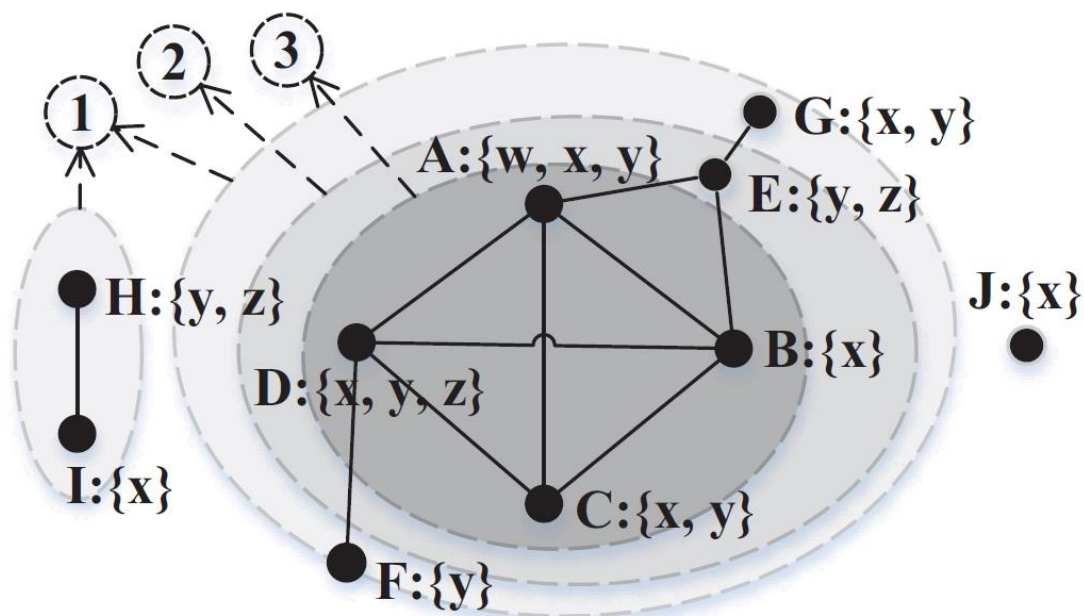


9.5.2 社团搜索

- 判断 $G_q(S')$ 是否存在^[13]
 - 先结构检查, 后关键字约束
 - 先关键字筛选, 后结构检查
 - 混合法
- 子图每个点 $d(v) \geq k+1$, 暗示 $d(v) \geq k$

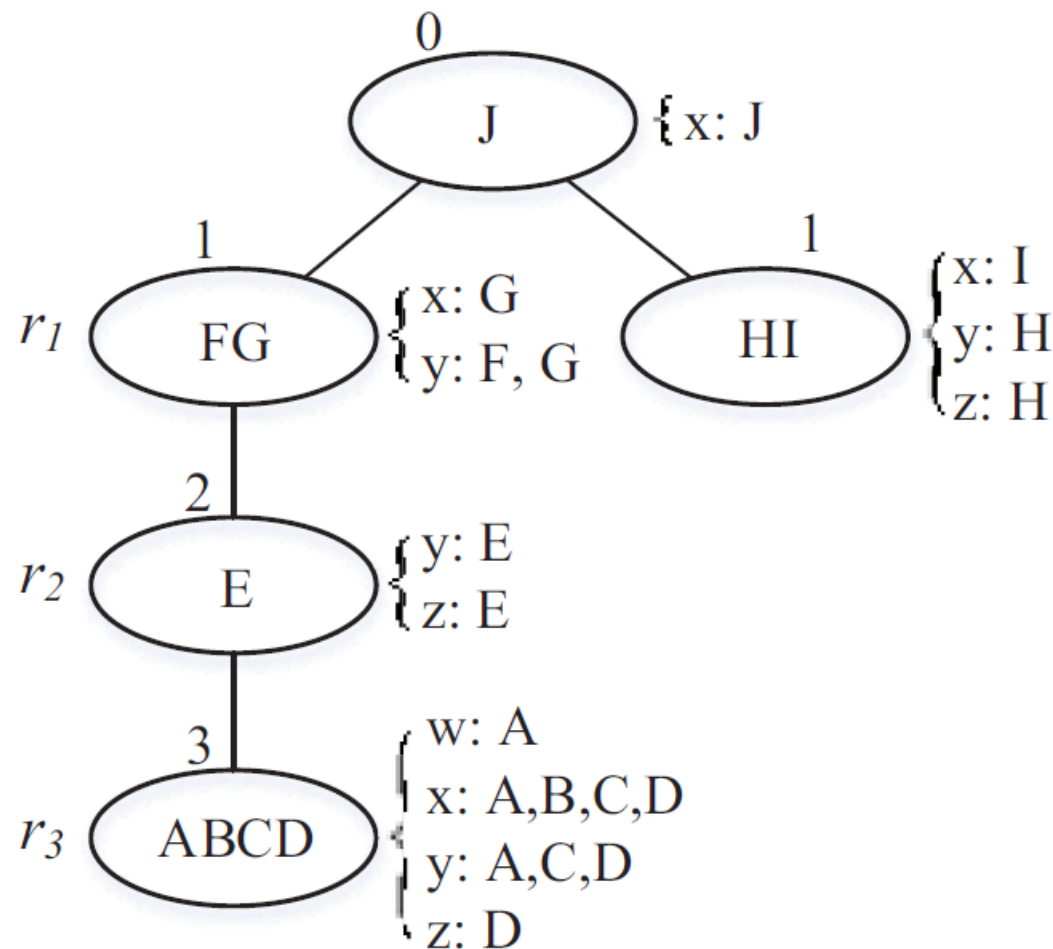
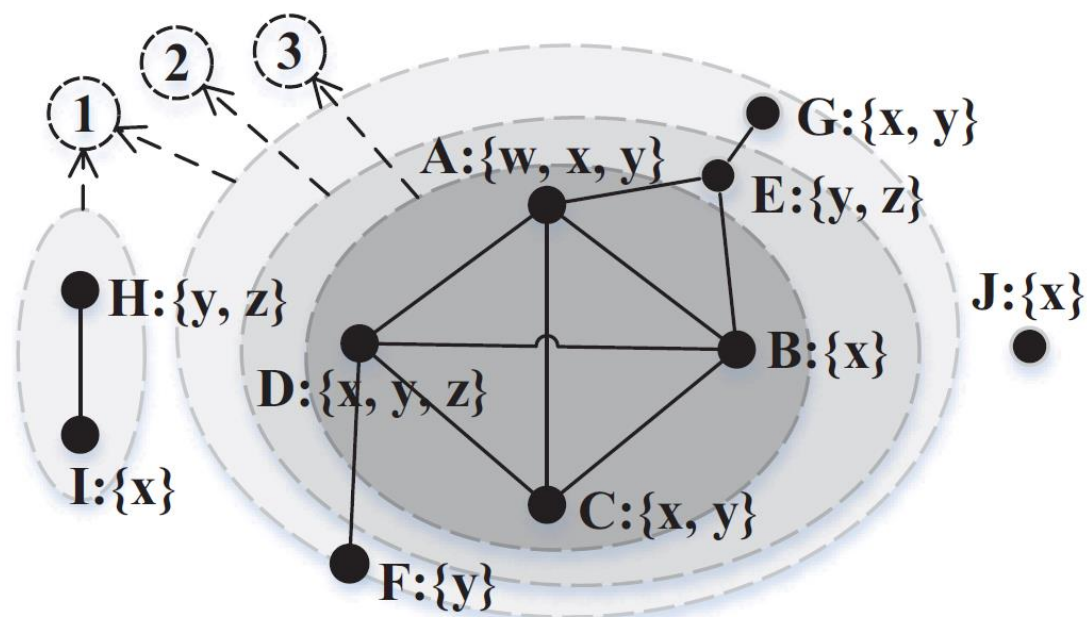
9.5.2 社团搜索

- 判断 $G_q(S')$ 是否存在^[13]



9.5.2 社团搜索

- 判断 $G_q(S')$ 是否存在^[13]



9.5.2 社团搜索

- 关键字查询

- 基于属性的社团查询

相同点：指定一些关键词或属性作为约束

(1) 不需要指定查询顶点

(2) 对密度无要求

(3) 无要求

(1) 给出查询的顶点，个性化的结果

(2) 密度有约束

- 每个顶点的度数不小某个数值

(3) 结果中每个顶点都包含给定的关键词约束

总结

- SPARQL语法
- SPARQL查询

查询语言：
SPARQL



- 基于本体的近似查询
- 融合结构和语义的近似查询

子结构查询



关键字查询



- 路径查询
- 社团搜索


其他查询



参考文献

- [1] https://en.wikipedia.org/wiki/Graph_isomorphism
- [2] Weiguo Zheng, Xiang Lian, Lei Zou, Liang Hong, Dongyan Zhao: Online Subgraph Skyline Analysis over Knowledge Graphs. IEEE Trans. Knowl. Data Eng. 28(7): 1805-1819 (2016)
- [3] Shan Y, Chen Y. Constructing target-aware results for keyword search on knowledge graphs[J]. Data & Knowledge Engineering, 2017, 110
- [4] Zheng W, Zou L, Peng W, et al. Semantic SPARQL similarity search over RDF knowledge graphs[J]. Proceedings of the Vldb Endowment, 2016, 9(11):840-851.
- [5] Julian R. Ullmann. An Algorithm for Subgraph Isomorphism[J]. J ACM. 1976, 23(1):31-42
- [6] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, Mario Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs[J]. IEEE Trans Pattern Anal Mach Intell. 2004, 26(10):1367-1372
- [7] Wu Y, Yang S, Yan X. Ontology-based subgraph querying[C] IEEE, International Conference on Data Engineering. IEEE, 2013:697-708.
- [8] S. Fortunato. Community detection in graphs. Physics Reports, 486(3):75-174, 2010.

参考文献



- [9] M. Newman et al. Finding and evaluating community structure in networks. Physical review E, 69(2):026–113, 2004.
 - [10] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In KDD, pages 939–948, 2010.
 - [11] W. Cui et al. Online search of overlapping communities. In SIGMOD, pages 277–288, 2013.
 - [12] W. Cui et al. Local search of communities in large graphs. In SIGMOD, pages 991–1002, 2014.
 - [13] Y. Fang et al. Effective community search for large attributed graphs. In PVLDB, pages 1233–1244, 2016.
- Algorithm for Matching Large Graphs[J]. IEEE Trans Pattern Anal Mach Intell. 2004, 26(10):1367–1372