# 2 相关技术现状

本章我们将介绍RDF作为知识的一种表示方式，在此基础上构建起来的Dbpedia知识库，然后介绍通用分布式运算框架Spark，最后一节将是经典的图频繁模式挖掘算法。

## 2.1 RDF知识描述与知识库

资源描述框架(Resource Descrption Framework, RDF)是W3C组织推荐的一个开放元数据框架，用来描述语义网资源及其之间关系。

关联数据的数据模型的核心是RDF数据模型，使用这一模型表示万维网资源，例如知识系统中的实体资源。之所以使用RDF作为知识库表示的描述方式是因为，它提供了应用程序之间的互操作性，可以在网络上交换“机器可理解”的信息。RDF数据模型强调网络资源的自动处理能力，例如：提供更好的检索能力，这一点使用SPARKQL成熟的系统可以实现，同时提供了知识共享和交换的能力，最重要的一点是提供了描述内容和内容之间关系的能力。

事实上，大量已有的知识库都建立在RDF数据模型的基础上，包括Freebase，YAGO，以及本文将要使用到的Dbpedia。

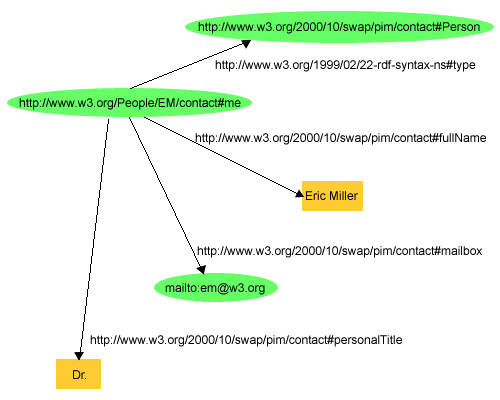
具体而言，RDF以Web标识符URI来标识资源（resource），并通过属性（property）和属性值来描述资源。资源是可拥有URI的任何事物，资源拥有相应的属性以及对应的属性值。

资源、属性和属性值三元组可形成一个RDF陈述（statement）。这三部分被称为陈述的主语（subject）、谓语（predicate）和宾语（object）。主语通常是指相应资源的URI引用（URI reference），谓语是表示某一属性的URI引用，而宾语是指属性值，它既可以是某一个资源的URI引用，也可以是一个字面量（literal），通常由值（value）和值的类型构成。主语和宾语可以是没有URI引用表示的匿名资源，也称为空白结点（blank node）。

由此，RDF三元组可形式化地定义为。其中，U是URI引用的集合，B是空白结点的集合，L是任意可能的数据类型所对应的字面量的集合。

一组RDF三元组定义为一个RDF图。RDF图被表示为一个有向标记图，其中节点可以为主语或者宾语，节点的表示引用可以用URI引用、字面量或者空白节点标记（空白结点标记通过程序唯一生成），有向边用谓语的URI引用标记，边的方向总是主语指向宾语，每条有向边以及它所连接的两个节点对应一个RDF三元组。

例如下图表示的是四个三元组构成的RDF图，其中subject只有一个，用URI标识为“http://www.w3.org/People/EM/contact#me”，而四个三元组分别描述了关于他的全名，类型，邮箱地址还有称呼的信息。



作为知识库中有代表意义的Dbpedia，在它的整个知识库建设上，是将维基百科中包含的内容转变为结构化的知识，并且在LOD原则指导下互联，将网上其它数据集和维基百科的数据结合起来，使得这些信息以关联数据的形式在Web上共享。Dbpedia数据集包括来自80多种语言的数据、400多万指向其它LOD数据集的联接和总共已超过20亿的RDF三元组，这些RDF三元组分为核心的DBpedia数据集和与外部资源链接的数据集。DBpedia的RDF三元组数据以文本文档的形式提供。

Dbpedia数据来源于Wikipedia，主要因为人在知识创造中具有不可替代的地位，为了提高数据集的语义处理能力,以人工方式建立了Dbpedia本体，避免计算机自动创建本体的局限性，支持用户对数据进行反馈和修改操作，增强计算机的语义处理能力；Dbpedia通过知识抽取框架自动抽取Wikipedia的标签、摘要、语言链接、图片等数据项,并进行一定的语义处理，以多元组的形式存放到知识库中，发挥了计算机对结构化程度较高的数据的重复操作能力。Dbpedia支持定时抽取和实时抽取两种抽取模式,前者以一个月为周期,而后者可以动态监测Wikipedia中的数据变化,进而保证Dbpedia与Wikipedia的同步性。

## 2.2 分布式运算框架Spark

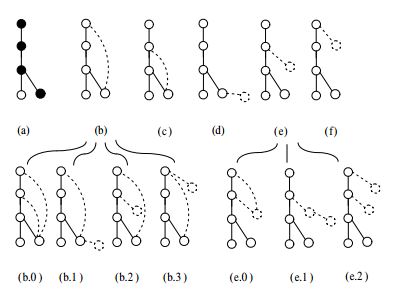
## 2.3 频繁模式挖掘算法

频繁模式（Frequent pattern）是频繁地出现在数据集中的模式（如项集、子序列或子结构）。例如频繁地同时出现在交易数据集中的商品（如牛奶和面包）的集合是频繁项集。子结构可能涉及不同的结构形式，如子图、子树或子格，它可能与项集或子序列结合在一起。如果一个子结构频繁地出现，则成为（频繁）结构模式，本文中正是需要使用到子结构即子图的频繁模式挖掘。

Apriori算法是R.Agrawal和R.Srikant于1994年提出的为布尔关联规则挖掘频繁项集的原创性算法。算法使用一种称作逐层搜索的迭代方法，K项集用于探索（K+1）项集。首先，通过扫描数据库，累积每个项的计数，并收集满足最小支持度的项，找出频繁1项集的集合。该集合称作L1。然后，L1用于找频繁2项集的集合L2，L2用于找L3，如此下去，直到不能再找到频繁K项集。找每个Lk需要一次数据库全扫描。但是显然在K项集寻找（K+1）项集的过程中需要扫描一次数据库，并产生大量的候选集，极大地降低了挖掘的效率。

从频繁项集的挖掘到频繁子图的挖掘，Apriori对应的算法是AGM、FSG和路径连接方法，同样当连接两个大小为K的频繁子结构产生大小为K+1的图候选时，基于Apriori算法的系统开销很大，并且类Apriori方法需要采用宽度优先搜索（BFS）。相反，模式增长算法在搜索方式上更加灵活，它可以使用宽度优先搜索，也能使用深度优先搜索（DFS），后者占用较少内存。PatternGrowthGraph算法便是其中的一种，该算法对于每个发现的图G，它递归地进行扩展，直到发现所有嵌入G的频繁图为止。然而该算法也有非常大的缺陷，它的瓶颈在于扩展图的低效率，由于同样的图可能重复发现多次。相同的图的重复发现导致计算的低效率。称二度发现的图为复制图（duplicate graph）。

为了减少复制图的产生，每个频繁图应该尽可能适当地扩展，在这样的思路下，gSpan算法应运而生，该算法也是基于模式增长算法，并引入DFSCode和最右扩展旨在减少复制图的产生，并且引入最小DFS编码来简化匹配过程，也就是说它不需要为检测复制来搜索先前发现的频繁图。同时，它不扩展任何复制图，但是仍保证发现频繁图的完全集。具体而言，遍历一个图有若干种可能，每一种遍历方式对应一棵DFS树，由于PatternGrowth方法在每个可能的位置扩展频繁图，这可能产生大量的复制图。然而gSpan算法引入了一种更为复杂的扩展方法。新的方法对扩展有如下限制：给定图G和G的DFS树T，一条新边e可以添加到最右顶点和最右路径上另一个顶点之间（后向扩展）；或者可以引进一个新的顶点并且连接到最右路径上的顶点（前向扩展）。进而把这个扩展过程定义两种类型的序：边序和序列序，基于上述规则的序称作DFS词典序（DFS Lexicographic Order）。如图所示



基于DFS词典序，很容易算出给定图G的最小DFS编码，因此对于同一幅图我们可以知道当前子图是否已经被搜索并挖掘，从而达到减少复制图产生的效果。

## 2.4 本章小结

本章我们简单介绍了关联数据中的知识表示方法以及作为知识库数据模型的RDF资源描述框架，关于S-P-O三元组的具体表示方法。然后我们介绍了Spark分布式运算框架，包括运行模式以及Map-Reduce编程范式。最后我们介绍了作为频繁模式挖掘基础的Apriori算法以及对应的频繁子图算法，并且分析了它们各自的优缺点。